UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

PROYECTO FINAL DE GRADO

Ingeniería Biomédica e Ingeniería Mecánica

VIRTUAL MIRROR

APLICACIÓN DE REALIDAD VIRTUAL

Memoria y Anexos

Autor: Cindy Paola Vega Pinzón
Oscar Mayorgas

Director: Jordi Torner
Co-Director: Francesc Alpiste

Convocatoria: Julio 2019

Resum

La realitat virtual és un entorn escènic el qual té com a objectiu projectar diferents escenaris d'aparença real utilitzant modelatge i simulació per ordinador. Aquest entorn permet la interacció de l'usuari mitjançant diversos dispositius sensorials, com auriculars, cascs, ulleres, guants o vestits per aconseguir una percepció de diferents estímuls que intensifiquin la sensació de realitat.

L'objectiu d'aquest projecte és dissenyar i implementar una aplicació mèdica amb el propòsit d'estudiar el comportament dels pacients amb mobilitat reduïda i l'anàlisi en temps real dels angles i moviments de les extremitats inferiors a partir de la captura de moviment del cos humà, i la representació gràfica d'un Avatar projectat en un mirall virtual; permetent d'aquesta manera, que a partir dels resultats obtinguts en l'aplicació dissenyada, el personal mèdic pugui analitzar de forma precisa l'exercici del pacient durant el seu procés de rehabilitació. Actualment, Les tecnologies de Realitat Virtual mostren potencial com eines de rehabilitació després d'un traumatisme i millores en comparació amb una fisioteràpia convencional.

Per complir amb els objectius establerts, es van utilitzar eines com 'Arduino', 'Unity', 'Kinect' les quals es van implementar amb l'objectiu de recopilar, processar i emmagatzemar les dades per a un estudi addicional que proporcioni resultats satisfactoris per al personal mèdic i permetés l'ús d'entorns multidimensionals interactius, beneficiós per a la recuperació dels pacients.

Els resultats del treball mostren l'eficiència de l'aplicació mèdica realitzada a causa de l'acceptació i viabilitat per part del personal mèdic, sent una alternativa didàctica, precisa, i de fàcil accés.

Resumen

La realidad virtual es un entorno escénico el cual tiene como objetivo proyectar diferentes escenarios de apariencia real utilizando modelado y simulación por computadora. Este entorno permite la interacción del usuario mediante diversos dispositivos sensoriales, como auriculares, cascos, gafas, guantes o trajes para una lograr una percepción de diferentes estímulos que intensifiquen la sensación de realidad.

El objetivo de este proyecto es diseñar e implementar una aplicación médica con el propósito de estudiar el comportamiento de los pacientes con movilidad reducida y el análisis en tiempo real de los ángulos y movimientos de las extremidades inferiores a partir de del cuerpo humano, y la representación gráfica de un Avatar proyectado en un espejo virtual; permitiendo de este modo, que a partir de los resultados obtenidos en la aplicación diseñada, el personal médico pueda analizar de forma precisa el desempeño del paciente durante su proceso de rehabilitación. Actualmente, las tecnologías de Realidad Virtual demuestran un alto potencial como herramientas de rehabilitación después de un traumatismo.

Para cumplir con los objetivos establecidos, se utilizaron herramientas como 'Arduino', 'Unity', 'Kinect', las cuales se implementaron con el objetivo de recopilar, procesar y almacenar los datos con el propósito de que el personal médico pudiera estudiar posteriormente los resultados registrados a partir del uso de entornos multidimensionales interactivos, beneficioso para la recuperación de los pacientes.

Los resultados del trabajo muestran la eficiencia de la aplicación médica realizada debido a la aceptación y viabilidad por parte del personal médico y del paciente siendo una alternativa didáctica, precisa y de fácil acceso.

Abstract

The virtual reality is a scenic environment, which aims to project different real-looking scenarios utilizing computer modelling and simulation. This environment allows for user interaction using various sensorial devices such as headphones, helmets, glasses, gloves or suits. Beforementioned tools receive or send information, which is a subject of further processing.

The objective of this project, is to design and implement a medical application for the purpose of studying the behavior of patients with reduced mobility and a real-time analysis of the angles and movements, of the lower extremities from the capture of movement of the human body, and the graphical representation of an Avatar projected into a virtual mirror allowing the medical staff accurately analyze the patient's performance during the rehabilitation process. Currently, Virtual Reality technologies demonstrate high potential as rehabilitation tools after trauma.

In order to meet the stated objectives, the tools like 'Arduino', 'Unity', 'Kinect' and `XAMPP server" were implemented with the goal of collecting, processing and storing the data for further study which would provide satisfactory results for medical personnel and would allow the usage of interactive multidimensional environments, beneficial both for patients and the medical staff.

The results of the work show the efficiency of the application, achieved by the implementation of an interactive and didactive system, enforced by its precision and the convenience of use.

Agradecimientos

En primera instancia, agradecemos a nuestros padres, por su apoyo, comprensión y motivación en cada uno de nuestros años de formación, a nuestros profesores por acompañarnos y brindarnos su conocimiento y bases fundamentales para poder formarnos como ingenieros, a cada una de las personas que nos estuvo acompañando en nuestro proceso formativo.

Agradecemos a la Universidad Politécnica de Cataluña por darnos la oportunidad de realizar nuestra investigación en esta institución y brindarnos todas las herramientas necesarias para poder llevar a cabo nuestros objetivos.

De igual manera, queremos agradecer a los profesores, Jordi Torner, Francesc Alpiste y Gil Serrancoli, por su acompañamiento durante el proyecto.

~ Cindy Vega y Oscar Mayorgas

Así mismo agradezco, a la universidad del Rosario y la Escuela Colombiana de Ingeniería Julio Garavito por acompañarme y brindarme la oportunidad de realizar una doble titulación y brindarme las bases fundamentales para ser una excelente profesional.

~ Cindy Vega

Índice

Índice de Figuras

Índice de tablas

1. PREFACIO

1.1. Origen del trabajo

A lo largo de los años, los avances tecnológicos y la medicina han ampliado su campo de conocimiento a través de la invención de nuevas técnicas, permitido crear nuevas investigaciones científicas, extensión de posibilidades de experimentación y adquisición de nuevos métodos.

Sin embargo, actualmente, el uso de este tipo de tecnologías no es accesible para todas las personas que lo solicitan, debido a su costo, ubicación, tiempo que requiere el personal y el paciente para el tratamiento y demanda del servicio.

Por este motivo se ve necesario el diseño de una aplicación médica dirigida a los pacientes con movilidad reducida; la cual comprenda un programa de rehabilitación específico de fácil acceso y uso, para el paciente y el terapeuta, teniendo en cuenta un entorno didáctico y un diagnóstico fiable portátil, para cualquier entorno.

El presente trabajo de grado es la continuación de un proyecto dirigido por el equipo de Jordi Torner, Francesc Alpiste y Gil Serrancolí. Fue realizado por los estudiantes Joel Guerrero Miranda i Aleix Santaeularia Riera. Así mismo, se realiza un proyecto paralelo con estudiante de la UPC, en el cual se estudia las extremidades superiores del paciente.

1.2. Motivación

La fuerza que nos impulsó a realizar este proyecto de grado, se evidencio en nuestro interés mutuo en las áreas de rehabilitación médica, informática y tecnologías biomédicas. Teniendo en cuenta, que uno de los objetivos iniciales y principales era lograr diseñar y desarrollar una aplicación médica que permitiera al usuario realizar el proceso de rehabilitación de la manera más fácil y entretenida.

Podemos señalar las principales razones de ello:

Uno de los principales factores es la repetitividad del procedimiento propuesto; después de cierta cantidad de tiempo, los pacientes pierden su interés en continuar el proceso debido a la falta de entretenimiento visual que desmotiva a los pacientes y los deja insatisfechos con el proceso de rehabilitación propuesto.

Por otro lado, notamos la necesidad de diseñar una plataforma que permita al personal médico almacenar, monitorear y analizar los datos de los pacientes de una manera más fácil y asertiva.

Así mismo, esperamos que nuestro trabajo contribuya a futuras investigaciones y nuevas versiones de este tipo de aplicaciones médicas y afecte positivamente la calidad de vida de las personas con discapacidades.

1.3. Requerimientos previos

Para poder alcanzar todos los objetivos propuestos dentro del presente proyecto de grado es necesario tener experiencia y conocimientos previos en el área de rehabilitación, programación, biomecánica y fisiología.

2. INTRODUCCIÓN

La rehabilitación es un proceso continuo y global realizado para impulsar, evaluar, tratar y alcanzar excelentes niveles de habilidades funcionales e independencia física de los pacientes. Les ayuda a lograr un alivio completo o una disminución de los síntomas tratables con el fin de permitir que los pacientes tengan una vida autónoma y dinámica. Existen tres indicadores principales de la recuperación motora que se basan en rutinas específicas, intervención temprana y la intensidad de la repetición.

El proyecto implica el desarrollo de una aplicación médica utilizando los entornos Kinect y Unity. Está dirigido a pacientes con discapacidades motoras, especialmente para aquellos con movilidad reducida en los extremos inferiores de su cuerpo.

El método propuesto es una forma efectiva de realizar los planes de rehabilitación de una manera entretenida y adecuada. También permite al personal médico monitorear el proceso de rehabilitación de los pacientes de una manera fácil, brindándole al especialista las herramientas necesarias para comparar y analizar la efectividad de las rutinas propuestas para cada uno de los pacientes. Se espera que el método propuesto supere los programas de rehabilitación tradicionales.

Todos los pasos que se realizaron en el proceso de diseño y desarrollo de la aplicación se describirán a lo largo del presente trabajo.

2.1. Objetivos del proyecto

2.1.1. Objetivo General

- Diseño y desarrollo de una aplicación médica a partir de la realidad virtual; orientado a pacientes con discapacidades motoras, especialmente con movilidad reducida en extremidades inferiores.

2.1.2. Objetivos Específicos

- Diseñar una herramienta de rehabilitación médica a partir de la implementación de métodos didácticos y cómodo acceso para el personal médico y el paciente.

- Monitorear los procesos de los pacientes de manera sencilla; comparando y analizando la efectividad de la rutina propuesta a partir de sus antecedentes y desempeño en tiempo real, a partir de la extracción y el estudio de parámetros biomecánicos.
- Desarrollar una interfaz para el usuario que le permita observar su rutina de ejercicios de forma correcta, evidenciando las falencias y adecuada ejecución de ellas.
- Realizar las primeras pruebas piloto en pacientes con la aplicación médica diseñada (versión beta)

2.2. Alcance del trabajo

El presente proyecto de Investigación busca diseñar e implementar una aplicación médica eficaz para pacientes con movilidad reducida en extremidades inferiores; la cual pretende ser una aplicación didáctica, accesible y cómoda, permitiendo de este modo, estudiar la evolución y comportamiento del paciente para un posterior diagnóstico, a partir del movimiento del usuario en tiempo real, teniendo en cuenta la realidad virtual como herramienta principal del desarrollo del proyecto.

Así mismo, el alcance trazado en el presente proyecto de grado fue llegar a realizar las primeras pruebas 'piloto' de la aplicación medica diseñada con pacientes en procesos de rehabilitación.

Teniendo en cuenta las investigaciones y los proyectos anteriormente realizados, el presente proyecto de grado inicia a partir de la programación de un avatar, el cual respondía a los movimientos del paciente al entrar en contacto con los puntos de referencia de la Kinect, donde se podían observar los ángulos de las extremidades superiores y sus valores actuales correspondientes realizado por estudiante de la Universidad Politécnica de Cataluña.

3. ESTADO DEL ARTE

3.1. Reality Virtual

3.1.1. Definición y elementos

La realidad virtual es un entorno escénico que apunta a proyectar diferentes escenarios de apariencia real utilizando modelado y simulación por computadora de forma digital. El usuario posee diferentes puntos de referencia de la posición y la ubicación del avatar dentro de este entorno, en el cual, se puede modificar la perspectiva y navegar libremente en un mundo artificial.

Los gestos permiten al usuario interactuar con el entorno mediante el uso de dispositivos interactivos, que envían y reciben información para su posterior procesamiento mediante diversas herramientas, como auriculares, cascos, gafas, guantes o trajes [6].

Para conseguir este tipo de sensación, se debe tener en cuenta:

- **Hardware:** Elementos que debe constituir un sistema informático para crear el entorno de realidad virtual.
- **Software**: Rutinas y programas que permiten realizar tareas específicas en un equipo, las cuales contiene información sobre las escenas que se deseen formar.
- **Pantalla:** Proyección y formación de la imagen en la cual el usuario ve la plataforma realizada.
- **Mandos**: Soporte físico con el objetivo de interactuar con la consola, ordenador y/o videojuego
- **Sensores:** Detección de los movimientos del usuario al entrar en contacto con la plataforma, dando la sensación de estar inmerso dentro del entorno de realidad virtual.
- Actualmente existen diversos accesorios que pueden ser de gran utilidad para dar un efecto más realista a la experiencia, como, por ejemplo, la simulación del movimiento, el viento o la humedad, determinadas en el mercado como tecnologías 4D.

3.1.2. Tipos de Realidad Virtual

Los tipos de tecnologías de los sistemas de realidad virtual se dividen en tres grupos, los cuales se encuentran clasificados teniendo en cuenta el nivel de inmersión del usuario:

3.1.2.1. Según el grado de inmersión

Puede clasificarse en tres tipos: inmersiva, semi-inmersiva y no inmersiva [21].

- **Realidad Inmersiva**

  El usuario se encuentra inmerso en la plataforma virtual, en la cual, pierde el contacto directo con la realidad y se sumerge en un mundo tridimensional; el usuario tiene la sensación de estar en un entorno real en el que no está físicamente presente.

Figura 1. Realidad Virtual Inmersiva

- **Realidad semi- Inmersiva o inmersiva de proyección**

  El usuario se encuentra dentro de un escenario de tres pantallas verticales y una pantalla ubicada en el suelo, formando un cubo, en el cual se proyectan diversas imágenes que permiten al usuario interaccionar y moverse de manera libre por el escenario formado.

Figura 2. Realidad Semi- Inmersiva o inmersiva de proyección

- **No inmersiva**

El usuario no se encuentra en una inmersión global, por lo cual, el usuario accede a través de ordenador o mando a el mundo virtual, realizando una interacción a través de un avatar y diversas herramientas como, el teclado, ratón o joystick a el escenario deseado. Este tipo de tecnología es adecuada para visualizaciones científicas y aplicaciones de entretenimiento.

Figura 3. Realidad Virtual no Inmersiva

3.1.2.2. Según el hardware

Teniendo en cuenta el hardware, se pueden catalogar teniendo en cuenta el uso de, pantallas VR y sistemas desktop, realidad virtual en segunda persona y sistemas de telepresencia y telerobótica [20]:

- **Pantallas de Realidad Virtual y Sistemas Desktop**

  Se presentan mundos bidimensionales y tridimensionales en pantallas 2D y 3D, permitiendo el movimiento independiente en los entornos de interés, en este tipo de realidad virtual se implementan herramientas tales como, guantes, cascos, HMD, entre otras.

- **Realidad Virtual en 2ª persona**

  El usuario interacciona directamente con el mundo virtual debido a que tiene control y noción de que se encuentra en él, dentro de la escena, este tipo de sistemas comprometen percepciones y respuestas en tiempo real a las acciones realizadas por el usuario.

- **Sistemas de telepresencia y telerrobótica**

  Permiten al usuario la manipulación de dispositivos y robots ubicados en localidades apartadas, con el objetivo de experimentar virtualmente la acción. Para cumplir con este propósito se hace uso de herramientas como, cámaras, dispositivos táctiles, micrófonos y elementos de retroalimentación ligados a elementos de control remoto teniendo en cuenta conexiones Wireless.

Figura 4. Sistema de telepresencia y telerrobotica

3.1.2.3. Según la interfaz del usuario

Teniendo en cuenta la interfaz del usuario de pueden evidenciarse a partir de, cabinas de simulación, sistemas de mapeo por videos, sistema de ventanas, cabinas de manipulación, sistemas de realidad mixta o sistemas de realidad virtual múltiple [14].

- **Cabina de manipulación**

  Tienen como objetivo la representación realista del entorno a partir de pantallas teniendo en cuenta el uso de una cabina, el cual ofrece un ambiente acorde con la situación en la cual se encuentra el usuario.

- **Sistema de mapeo de video**

  Tiene como propósito la grabación del usuario a partir de cámaras de video e incorporar las imágenes en la pantalla del ordenador, permitiendo al usuario interactuar en tiempo real con otros usuarios o imágenes gráficas de descendencia virtual.

- **Sistemas de ventanas**

  Sistemas no inmersivos los cuales emplean una pantalla con el objetivo de observar un mundo digital, en la cual tiene como objetivo tener cualidades similares a la realidad.

- **Sistemas de Realidad Mixta**

  Se puede describir como el acoplamiento entre la Realidad Virtual y los Sistemas Telepresenciales, en el cual se incrementa la percepción del usuario respecto al mundo real mediante HMD.

- **Sistemas de Realidad Virtual Múltiple**

  El usuario evidencia el acoplamiento de estímulos auditivos, táctiles, visuales y de movimiento a partir del uso de inteligencia artificial.

3.1.3. Áreas de aplicación

Específicamente, la realidad virtual tiene diversas aplicaciones en diferentes disciplinas, de la cuales se destaca, la medicina, entretenimiento, educación, turismo, museografía, y muchas otras. En este

proyecto, la realidad virtual se concentró en un entorno médico, en el que se estudió el comportamiento del cuerpo humano.

La solución propuesta es útil en el campo de la medicina, ya que facilita el proceso de enseñanza y capacitación de los médicos. Permite el modelado y la programación en 3D para observar y estudiar la anatomía y las patologías del paciente en particular, mediante el uso de nuevas tecnologías biomédicas para intervenciones quirúrgicas, diagnósticos de imagen, rehabilitación y estudios fisiológicos.

Así mismo, actualmente existen diversas investigaciones en el área de la salud, las cuales tienen como objetivo combatir las enfermedades como: Alzheimer, ansiedad, hiperactividad, autismo, síndrome de asperger, déficit de atención, obesidad, fobias y traumas psicológicos se ha llevado a cabo de la misma manera [16].

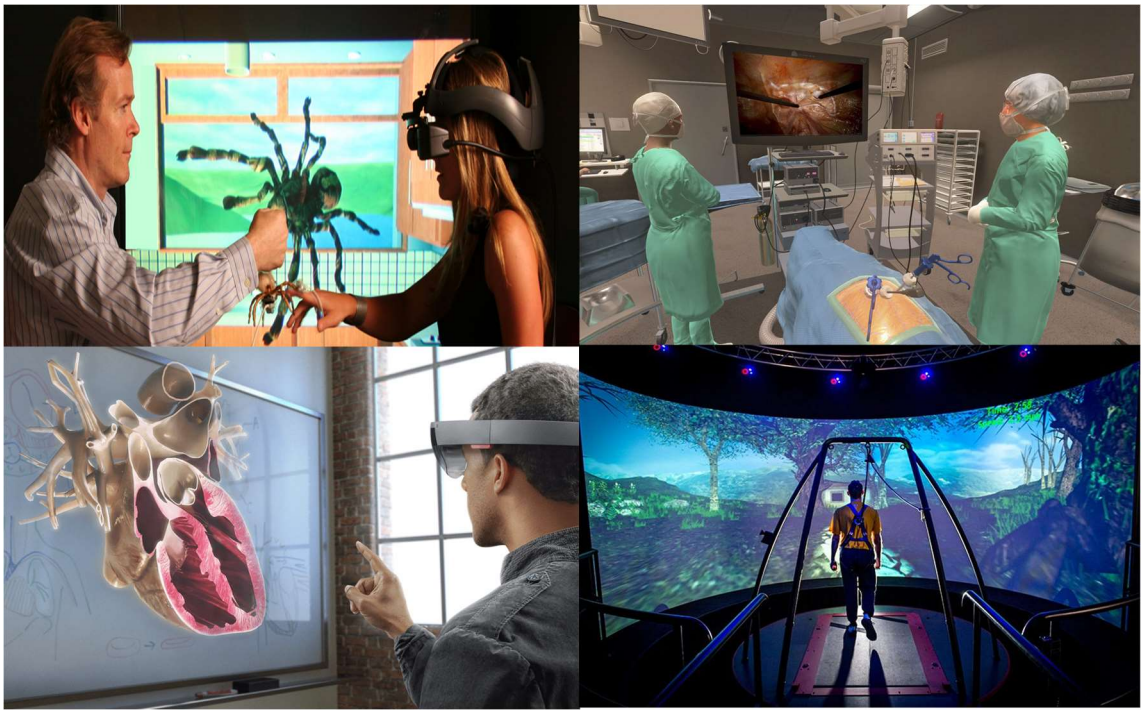


Figura 5. A) Aplicación médica con el objetivo de contrarrestar fobias B) Simulación de quirófanos con propósito quirúrgico C) Realidad Virtual y aumentada para el estudio del cuerpo humano D) Aplicación médica en el entorno de rehabilitación médica

3.1.4 Estudio sobre la eficacia de la Realidad Virtual en la rehabilitación

Teniendo en cuenta estudios, pruebas y análisis previamente realizados por centros de investigación, compañías y otras entidades, está claro actualmente que la tecnología médica va incrementando a través del tiempo, en el cual los procesos de rehabilitación se ven involucrados y van acoplando nuevas tecnologías a sus rutinas.

Para sustentar lo anteriormente mencionado, se describirá un estudio recientemente realizado respeto a los resultados motores, cognitivos y funcionales de pacientes adultos con accidentes cerebrovasculares en contacto con la rehabilitación virtual.

El estudio precisa que las tecnologías de realidad virtual muestran la efectividad de los procesos de rehabilitación después de un traumatismo. En particular, el sistema Elements, contiene un sistema de computación de superficie personalizada e interfaces tangibles; el cual, va enfocado a tratamiento para la función cognitiva y de las extremidades superiores luego de una lesión cerebral traumática [17].

El método fue realizado con 21 adultos entre 42 y 94 años con accidente cerebrovascular subagudo, en el cual, se asignó al azar tres sesiones semanales entre 30 y 40 min combinadas con el tratamiento habitual con ayuda del fisioterapeuta y autónomo. La habilidad de los miembros superiores (Prueba de caja y bloques), cognición (Evaluación cognitiva de Montreal y subpruebas seleccionadas de CogState), y la participación diaria (Inventario de funcionamiento neuroconductual) se examinaron antes y después del entrenamiento de pacientes hospitalizados, y un mes después.

Con el entrenamiento de Elements se pudo demostrar las mejoras de la función motora de las extremidades estudiadas. La recuperación proporcional fue de dos a tres veces mayor que la de los participantes de recuperación habitual, con una transferencia superior a las conductas motrices, cognitivas y de comunicación cotidianas.

Un curso de rehabilitación virtual de Elements con tareas de movimiento de las extremidades superiores dirigidas y exploratorias facilita la recuperación cognitiva y motora después del accidente cerebrovascular. La magnitud de los efectos del entrenamiento, el mantenimiento de las ganancias en el seguimiento y la generalización de las actividades diarias brindan evidencia preliminar convincente del poder de la rehabilitación virtual cuando se aplica de manera específica y basada en principios.

3.2. Unity

Programa creado y diseñado por Unity Technologies es un motor de videojuego multiplataforma, quiere decir que permite la creación de aplicaciones para múltiples plataformas a partir de un único desarrollo.

Su funcionamiento se basa en la creación de niveles de juego, conocidas como escenas dentro de Unity 3D. Cada escena se encuentra compuesta por una jerarquía de objetos (GameObject), los cuales están definidos por una serie de componentes, que son: su posición en las tres dimensiones, su escala y su rotación respecto a los 3 planos. Además, brinda la posibilidad de añadir diferentes comportamientos o funcionalidades a los objetos a través de la creación de scripts o funciones informáticas.

3.2.1. Entorno de Unity

Podemos dividir el editor de Unity en 5 vistas principales:

1. **Explorador**: Muestra un listado que contiene todos los elementos del proyecto, con el objetivo de hacer más fácil la forma de ordenar la aplicación. En esta ventana es posible encontrar las imágenes, scripts, escenas, audios, texturas, prefabs, atlas y todo el conjunto de los elementos que es posible utilizar en la aplicación.
2. **Inspector:** Permite la definición de las características de los elementos constituyentes del proyecto. Además, facilita la modificación de los valores de forma rápida, cambia texturas arrastrando ficheros desde el explorador, añade scripts.
3. **Jerarquía:** Lista estratificada de los elementos de la escena.
4. **Escena:** Edición del diseño de la aplicación completa o parte de la misma. La aplicación puede contener diferentes escenas que conjuntamente formarán los distintos niveles.
5. **Juego:** Permite la visualización de la aplicación a distintas resoluciones.

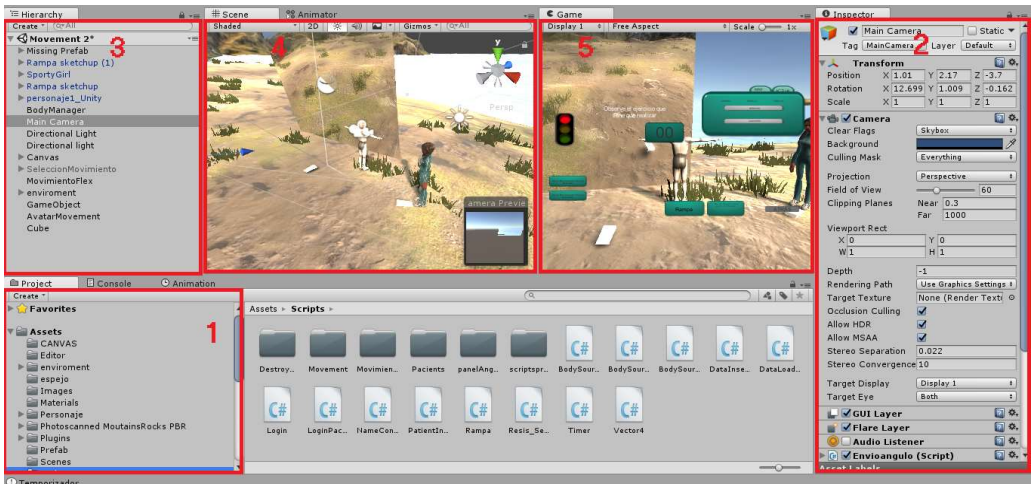


Figura 6. Entorno de Unity

3.2.2. Requerimientos del sistema para Unity

Componente Requisitos mínimos del sistema.

- Procesador: SSE2 (Streaming "Single Instruction Multiple Data" Extension 2)
- GPU: DirectX 10
- Sistema operativo: Windows 7 SP1, Windows 8, Windows 10 (versiones de 64 bits), macOS 10.11+

3.3 Microsoft Visual Studio

Las últimas versiones de Unity traen consigo un software de programación denominado Microsoft Visual Studio, este Software, es un entorno de desarrollo integrado (IDE) para sistemas operativos Microsoft. Tiene como objetivo, desarrollar programas informáticos de Microsoft Windows, así como sitios web, aplicaciones web, servicios web, y aplicaciones móviles. Este programa utiliza plataformas de desarrollo de software de Microsoft como Windows API, Windows Forms, Windows Presentation Foundation, Windows Store y Microsoft Silverlight.

Así mismo, es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc. En el presente proyecto de grado se hizo uso del lenguaje de programación C#, el cual es un lenguaje de programación orientado a objetos.

3.4. HTC Vive

3.4.1. Definición y características

HTC Vive hace referencia a una tecnología de visualización a partir de gafas específicamente para realidad virtual fabricadas por HTC y Valve, diseñadas para sumergirse en un mundo virtual en el que se permite al usuario realizar diferentes actividades utilizando controladores para interactuar con objetos virtuales.

Por medio de su sistema de tecnología de seguimiento llamado "lighthouse", los usuarios pueden moverse por un área de juego con su movimiento real reflejado en el entorno de realidad virtual e interactuar con él. A diferencia de una experiencia VR estacionaria, le permite al usuario navegar a través del espacio 3D sin usar un joystick.

Hay dieciséis componentes diferentes en la caja de HTV Vive, y a continuación explicaremos las funciones y propiedades de los más importantes:

- El Vive Headset se encuentra formado por dos paneles OLED con una resolución de pantalla de 1080 x 1200 x (2160 x 1200), una frecuencia de 90 Hz y un campo de visión de 110 grados. Así mismo, incluye una cámara frontal que brinda la posibilidad de observar el entorno sin quitarse los auriculares. El dispositivo consta de 36 sensores infrarrojos alrededor del auricular para determinar la ubicación del auricular en el espacio 3D en tiempo real. Para hacer la experiencia más segura, incluye un sensor de proximidad para proteger a los usuarios de obstáculos o paredes reales y posee una mayor precisión.
- El cuadro de enlace de Vive, es el enlace entre la computadora y las gafas Vive. Este tiene un puerto HDMI, un puerto de alimentación y un puerto USB, los cuales se utilizan para conectar el auricular mediante un cable 3 en 1. En el otro lado del dispositivo, contiene un puerto de alimentación y un puerto USB, un puerto HDMI y un Mini DisplayPort.
- Las estaciones base Vive son cajas negras de sincronización inalámbrica las cuales son usadas para el rendimiento del sistema de seguimiento Lighthouse. Estas, son colocadas a una altura mínima de 2 metros con una inclinación de 30 o 45 grados y una distancia de menos de 5 metros entre ellos. Al usar una dimensión mínima de 2 x 1.5 para la sala, las estaciones base Vive pueden crear un espacio virtual 3D. En el mismo periodo de tiempo, se ejecuta una aplicación de experiencia en realidad virtual, los sensores infrarrojos de los auriculares detectan con precisión submilimétrica los pulsos infrarrojos temporizados que emiten las estaciones base a 60 pulsos por segundo.
- Los controladores Vive son receptores similares a las gafas Vive y su ubicación se determina a través de veinticuatro sensores infrarrojos que están situados en su anillo. Ambos

controladores tienen múltiples métodos de entrada para cargar, un disparador de doble etapa, botones de agarre y un track pad.

Los accesorios descritos a continuación, que son material de soporte, completan el paquete HTC Vive:

- Adaptador de corriente de caja de enlace
- Adaptador de corriente de la estación base (2 unidades)
- Auriculares
- Cable de sincronización
- cable HDMI
- cable USB
- Cargador micro USB (2 unidades)
- Cojín facial alternativo
- Documentación
- Kit de montaje
- Paño de limpieza.
- Plataforma de montaje de caja de enlace

3.4.2. HTC Vive sistema de seguimiento

A partir del sistema de tecnología de seguimiento llamado "Lighthouse", los usuarios pueden moverse por un área de juego con su movimiento real reflejado en el entorno de realidad virtual e interactuar con él. A diferencia de una experiencia VR estacionaria, le permite al usuario navegar a través del espacio 3D sin usar un joystick.

3.4.3. HTC Vive requerimientos del sistema

A continuación, describimos los requisitos del sistema de nuestro PC que necesita para disfrutar de la experiencia VIVE.

- GPU: NVIDIA GeForce GTX 970, AMD Radeon R9 290 o equivalente
- La salida de vídeo: HDMI 1.4, DisplayPort 1.2
- Memoria 4 GB RAM
- Procesador: Intel Core i5-4590 / AMD FX 8350 o equivalente
- Puerto: USB 1x USB
- Sistema operativo: Windows 7 SP1, Windows 8.1, Windows 10

3.4. Arduino

Es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines hembra, los que permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla. El lenguaje de programación de Arduino está basado en C++ [2].

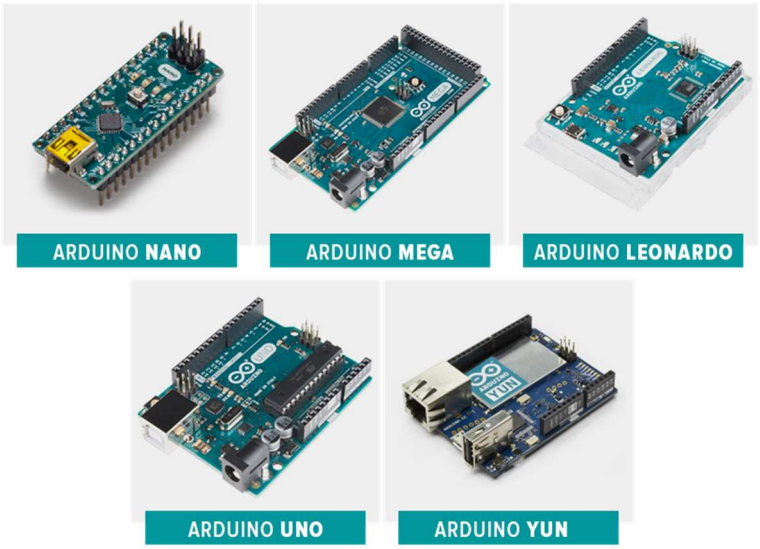


Figura 7. Tipos de Arduinos en el mercado.

3.4.1. Arduino UNO

3.4.1.1 Especificaciones técnicas

El Arduino UNO tiene las siguientes especificaciones:

- Microcontrolador ATmega328.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad
- Voltaje de entrada 7-12V.

3.4.1.2 Requerimientos del sistema

Para poder programar la tarjeta es necesario el entorno de desarrollo Arduino, el cual lo podemos descargar en el siguiente link gratuitamente: https://www.arduino.cc/en/Main/Software#. Se puede descargar para los sistemas operativos Windows, Mac OS y Linux

3.4.2 Galga Extensiométrica

Es un sensor que nos permite medir la presión o deformación y se basa en el efecto piezorresistivo. Su forma más común consiste en un estampado de una lámina metálica fijada a una base flexible y aislante.

3.5. KINECT

Kinect es un dispositivo principalmente fabricado para la Xbox 360, y desde el 2011 para PC, desarrollado por Microsoft, que permite a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional (como podría ser el mando o el teclado), mediante el reconocimiento de gestos, objetos, imágenes, y los comandos de voz.

3.5.1 Especificaciones técnicas

Para detectar el movimiento y crear imágenes físicas en la pantalla, Kinect dispone de tres piezas que trabajan juntas: una cámara de video VGA (Video Graphics Array) en color RGB (rojo, verde y azul), un sensor de profundidad y un micrófono de múltiples matrices.

- Cámara de video en color VGA - Esta cámara de video detecta la cara detectando tres colores: rojo, verde y azul.
- Micrófono de matriz múltiple: cuatro micrófonos que pueden escuchar las voces de los jugadores y evitar el ruido externo.
- Sensor de profundidad: un proyector IR y un sensor CMOS monocromático trabajan juntos para "ver" la habitación en 3D independientemente de las condiciones de iluminación.

3.5.2 Requerimientos del sistema

Sistema operativo soportado: Estándar incrustado 8, Windows 8, Windows 8.1 (o posterior)

Configuración de hardware mínima:

- GPU DirectX 11
- Memoria 2 GB RAM
- Procesador de 64 bits (x64), doble núcleo físico a 3,2 GHz
- Puerto USB 3.0

3.6. Rehabilitación

3.6.1. Tipos de rehabilitación médica relacionadas con el proyecto

- **Rehabilitación Ortopédica:** Es la ciencia que proporciona tratamiento médico rehabilitador a pacientes con déficit o alteraciones del sistema músculo esquelético o articular [22].
- **Rehabilitación Neurológica:** Hace referencia al tratamiento médico de rehabilitación que se ofrece a pacientes que han sufrido una lesión del sistema nervioso central o periférico con la finalidad de reintegrar al individuo a la sociedad [22].

3.7. Grados de discapacidad

3.7.1 Definición y tipos clasificación

Según la OMS (Organización Mundial de la Salud), se considera como discapacidad cualquier tipo de restricción o dificultad para realizar cualquier actividad de forma normal. Una discapacidad puede ser provocada por un exceso o déficit de función. Además, podemos subclasificar la discapacidad según sea temporal o permanente en el tiempo.

Mediante una valoración física, sensorial y biopsicosocial es posible determinar el grado de discapacidad de un individuo de forma objetiva. El grado de discapacidad se expresa en forma de porcentaje y responde a términos técnicos unificados y establecidos por la administración pública.

3.8. Contexto de aplicación realizada

El presente proyecto va dirigido a mejorar la motricidad y la biomecánica de las extremidades inferiores de pacientes que padecen un déficit o alteración de su función. Así pues, los sujetos con movilidad reducida pueden beneficiarse de la presente aplicación debido a su forma didáctica y de su sencillo acceso. Un ejemplo concreto de la terapéutica rehabilitadora que supone el ejercicio

que proponemos, sería el reinicio de la marcha en pacientes que han visto mermada su capacidad para caminar, sea cual fuese la causa de la misma.

3.9. Associació de Disminuïts Físics d'Osona

ADFO es una asociación formada por profesionales de carácter multidisciplinar, la cual tiene como objetivo proporcionar asistencia social, orientación y asesoramiento a personas discapacitadas y a sus familias.

Así mismo, proporciona un servicio de inserción laboral, de acompañamiento y soporte. Cabe destacar la labor de la organización en materia de fisioterapia neurológica dirigida a pacientes adultos, con daño cerebral adquirido o enfermedades neurodegenerativas. La entidad proporciona un diagnóstico fisioterápico en neurología, valoración pronostica funcional y planificación del tratamiento según los objetivos del paciente y la familia, a nivel físico, lingüístico y visual [18].

4. DEFINICIÓN DEL PROYECTO

4.1 Descripción del proyecto

El presente proyecto de grado tiene como objetivo desarrollar, diseñar e implementar una aplicación médica de realidad virtual precisa la cual proporcione acompañamiento a pacientes con movilidad reducida en su proceso de rehabilitación.

La aplicación médica diseñada está formada por tres escenarios interactivos. En el primer escenario, se registran los datos de los pacientes estudiados. Seguidamente, aparece un escenario que muestra las instrucciones referentes a la realización del ejercicio propuesto. Concretamente, se detalla una secuencia de ejercicios específicos de miembros inferiores y, por último, un escenario didáctico en el cual, el paciente deberá llevar a cabo una rutina de ejercicios siguiendo las indicaciones del modelo programado.

Un Avatar, nombrado SportyGirl (ver recuadro 1 de la Figura 31), reproducirá el ejercicio propuesto para mostrar al paciente la práctica del ejercicio, y consecutivamente el paciente deberá repetirlo. Es posible realizar el ejercicio con el uso complementario de las gafas VR o sin ellas. En el caso de realizarlo sin las gafas, se podrá observar la escena en tercera persona y se observara los movimientos reflejados en un avatar secundario (recuadro 2 de la Figura 31) . Si se implementan las gafas, la aplicación se realizará en primera persona y solo se podrá observar mediante un espejo situado en una zona próxima al paciente.

A partir de este movimiento, se evaluará el tiempo implementado, las repeticiones, los ángulos y flexiones de la parte del cuerpo de interés, para su posterior estudio por parte del personal médico.

Así mismo, la aplicación consta de un segundo nivel en el cual se estudia los miembros superiores del paciente, este nivel fue llevado a cabo por alumnos de la institución.

Con el propósito de cumplir con todos los parámetros estipulados y hacer asequible la aplicación, se diseñó un manual de instrucciones sobre el manejo de la aplicación (ANEXO A. Instrucciones Manejo de aplicación) en el cual se describen todos los pasos a tener en cuenta al interactuar con la plataforma de manera sencilla.

Para el correcto diseño de la aplicación médica, el presente trabajo se dividió en cinco fases; en las cuales se realizó un estudio de antecedentes, fundamentos básicos y teóricos requeridos para desarrollar correctamente el proyecto, en el que se tuvo en cuenta la identificación de parámetros fisiológicos, tales como, ángulos y movimientos del cuerpo humano.

Consecutivamente, se realizó la integración y modelado en 3D de la anatomía del cuerpo humano a través de un avatar en Unity, donde se creó el entorno, la estructura y botones de mando de las rutinas de ejercicio con el objetivo de hacer una plataforma más interactiva y didáctica.

Lo anteriormente descrito, se realizó a partir de la programación de un personaje en primera persona que interactuara con el entorno, un avatar orientador y los objetos presentes en el, tanto internos como externos. De igual manera, se programó el nivel de miembros inferiores con el objetivo de poder seleccionar el ejercicio de interés acorde a la rutina del paciente y estudiar los parámetros estipulados, comparando los resultados obtenidos y el avance que tiene el paciente dentro de su proceso de rehabilitación mediante el lenguaje de programación C#.

Para que esta aplicación fuese óptima para el personal médico y seguir el proceso de rehabilitación detallado del rendimiento del paciente, fue necesario implementar una base de datos, la cual es un sistema de gestión el cual tiene como objetivo almacenar información del paciente para poder ser analizada posteriormente por el cuerpo médico.

Igualmente, se requirió el uso de un hardware específico para la visualización de elementos virtuales dentro de la aplicación desarrollada, esto fue posible gracias a la empresa colaboradora *Visyon 360,* la cual proporcionó unas gafas de realidad virtual *HTV Vive* y un ordenador eficaz para el desarrollo de la aplicación.

Finalmente, se realizaron estudios y pruebas con pacientes y se compararon los objetivos propuestos inicialmente con los resultados obtenidos al observar la interacción entre la aplicación diseñada, el cuerpo médico y los pacientes a partir de la interacción y el estudio del comportamiento del paciente frente a las rutinas de rehabilitación de extremidades inferiores.

4.2. Planificación del proyecto

A continuación, se describe la planificación empleada para cumplir con los objetivos propuestos en el proyecto, con el propósito de realizar las tareas de forma organizada y secuencial.

4.2.1. Plan de Trabajo.

El proyecto se dividió en cinco fases, en las cuales se tuvo en cuenta el orden de prioridad y la duración de la actividad propuesta.

Primera fase - Asignación del proyecto:

- Asignación del proyecto VIRTUAL MIRROR y decisión de objetivos
- Indagación de los antecedentes del proyecto e investigaciones relacionadas.
- Estudio y primera revisión de fundamentos teóricos y herramientas complementarias para la comprensión y desarrollo del proyecto: programas informáticos (Curso de Unity Online) y conceptos básicos.

Segunda fase- Planificación del proyecto:

- Construcción de plan de trabajo y herramientas de trabajo grupal para un desarrollo óptimo (Metodología, Diagrama de Gantt y refuerzo de conocimientos).

Tercera fase - Fundamentos teóricos y prácticos:

- Segunda revisión bibliográfica (Marco teórico)
- Correcto uso de herramientas informáticas para llevar a cabalidad el proyecto (Unity, Kinect y Arduino)
- Revisión de los objetivos y falencias actuales de los programas de rehabilitación (Diálogo con fisioterapeutas y necesidades específicas por parte del usuario)

Cuarta fase - Diseño

- Reconocimiento de extremidades y articulaciones del cuerpo humano con el objetivo de establecer el movimiento de los pacientes y el avatar en tiempo real.
- Diseño de algoritmos para la medición de los ángulos de las extremidades inferiores a partir de Unity, Kinect y visual Studio.
- Medición de parámetros fisiológicos a partir de la interacción entre el individuo de prueba y el avatar.
- Implementación de programa en Arduino con el objetivo de medir la interacción entre el paciente y material de rehabilitación (Rampa)
- Diseño de interfaz y herramientas didáctica con el objetivo de observar los resultados de los pacientes.

Quinta fase - Aplicación

- Revisión y comparación de la aplicación final con los objetivos propuestos (Correcciones, características específicas por parte del personal médico, entre otras)
- Evaluación de los resultados obtenidos y viabilidad del proyecto.
- Memoria
- Entrega final del proyecto de grado.

Así mismo, se han realizado reuniones diarias con el objetivo de evaluar el progreso y el enfoque del proyecto teniendo en cuenta los comentarios realizado por los fisioterapeutas, tutores y expertos en el área. Igualmente, se realiza simultáneamente la memoria del proyecto en cada una de las fases estipuladas.

4.2.1. Diagrama de Gantt

A continuación, se describen las actividades realizadas teniendo en cuenta la duración de cada una de las tareas demandadas, con el objetivo de cumplir con los objetivos propuestos en el presente proyecto de grado [Anexo C. Planificación del proyecto – Diagrama de Gantt]

VIRTUAL MIRROR	INICIO	FINAL
ETAPA 1 - Asignación de Proyecto	**03/01/2019**	**31/01/2019**
Asignación del proyecto de grado	03/01/2019	03/01/2019
Antecedentes del proyecto y investigaciones relacionadas	03/01/2019	14/01/2019
Adquisición de fundamentos y herramientas informáticas	03/01/2019	31/01/2019
Revisión de fundamentos teóricos	14/01/2019	31/01/2019
Redacción de la memoria	03/01/2019	30/05/2019
ETAPA 2 - Planificación del Proyecto	**01/02/2019**	**01/04/2019**
Construcción del plan de trabajo y herramientas grupales	01/02/2019	08/02/2019
Marco teórico	08/02/2019	01/04/2019
ETAPA 3 - Fundamentos teóricos y prácticos	**15/02/2019**	**20/05/2019**
Revisión de objetivos C.Medico diseñados y descripción de la aplicación	15/02/2019	15/02/2019
Interacción con Unity, Kinect y Arduino enfocada a aplicación	15/02/2019	20/05/2019
ETAPA 4 - Diseño	**01/03/2019**	**20/05/2019**
Sincronización de avatar y articulaciones del cuerpo humano	01/03/2019	06/03/2019
Evaluación de movimientos en tiempo real (Avatar- Paciente)	06/03/2019	09/03/2019
Diseño de algoritmos	01/03/2019	20/05/2019
Medición de parámetros fisiólogicos (Avatar - Paciente)	09/03/2019	23/03/2019
Implementación de rutina de ejercicios	25/03/2019	08/04/2019
Estudio,diseño y adición de Rampa de Rehabilitación	25/03/2019	22/04/2019
Diseño de interfaz y herramientas interactivas	25/03/2019	20/05/2019
Implementación de base de datos	25/03/2019	20/05/2019
Implementación de HTC VIVE	01/05/2019	04/05/2019
Pruebas y revisión interna final	10/05/2019	25/05/2019
ETAPA 5 - Aplicación	**20/05/2019**	**08/07/2019**
Revisión y comparación con los objetivos de la aplicación	20/05/2019	22/05/2019
Evaluación y pruebas de la aplicación en contacto con pacientes	23/05/2019	23/05/2019
Entrega final de la memoria del proyecto	30/05/2019	04/06/2019
Defensa del proyecto de grado	25/06/2019	08/07/2019

Figura 8. Diagrama de Gantt Implementado en el diseño de la aplicación.

5. METODOLOGÍA

5.1 Elección del Hardware

La realidad virtual es un entorno generado a partir de la proyección de diversos escenarios teniendo en cuenta la simulación y el modelado por computadora a través de un Hardware, el cual tiene como objetivo crear una sensación de inmersión dentro de la plataforma utilizada; presenta una interacción, retroalimentación y visualización del entorno generado. Para cumplir con este propósito, actualmente se utilizó la tecnología HTC Vive.

HTC Vive, estimula al usuario, a partir de los sentidos con el objetivo de introducirlo en un mundo virtual considerando controladores de interacción con el propósito de interactuar con objetos virtuales; así mismo los sensores del hardware utilizado en la presente investigación poseen mayor grado de libertad de movimiento, por lo cual, puede proporcionar una mejor experiencia cuando el paciente se encuentre en contacto con la aplicación [24].

Figura 9. HTC Vive ("Tipo de gafas de realidad virtual" n.d.)

Para la elección de las HTC Vive, hemos tenido en cuenta otras gafas como las Oculus rift, Razer OSVR, PlayStation VR y las Samsung Gear VR. Las Samsung Gear fueron descartadas ya que solo pueden ser utilizadas con teléfonos móviles, y las PlayStation también porque solo se pueden utilizar en su propia plataforma. De las tres gafas restantes, las Razer OSVR eran las únicas que no disponían de más controladores que las propias gafas. Aunque en esta versión de la aplicación solo utilizamos las gafas, creemos que en un futuro podría ser útil otros controladores como los mandos de las HTC Vive. Las propiedades entre las HTC Vive y las Oculus Rift son prácticamente las mismas. Nos acabemos decantando por las HTC ya sus sensores captan más área.

5.2. Elección del Software.

El software implementado para el diseño de la aplicación médica fue Unity, debido a que sus características y acople a el diseño de la aplicación;

A continuación, se describen las principales características por las cuales se ha elegido Unity como plataforma principal:

- Aplicación gratuita disponible para Windows.
- Fácil integración con diferentes Softwares, programas de diseño y Hardware.
- Compatibilidad para desarrollo de realidad virtual dentro de la plataforma.
- Software gratuito de sencillo manejo.
- Herramientas y paquetes con el objetivo de ayudar a el usuario a dar solución durante la interacción con la plataforma.

Figura 10. Unity.

Otras aplicaciones de desarrollo podrían ser Game maker y Unreal Engine. Principalmente se ha optado por Unity 3D ya que es compatible con HTC Vive, las demás no disponen de una versión gratuita, y tiene una fácil integración de objetos modelados en otros programas (como es en nuestro caso Sketchup).

5.3. Software de programación

El software escogido para realizar la programación requerida en la presente aplicación médica, fue Microsoft Visual Studio, el cual es:

- Autocompletado inteligente
- Programa de fácil acceso y uso
- Software gratuito
- Compatibilidad con diferentes tipos de plataformas.
- Descripciones emergentes entre los elementos del código.
- Ofrece apoyo en la sintaxis del diseño de la aplicación.
- Esquematización de código realizado.

La elección del Software de programación en C# fue Visual Studio ya que es el que viene por defecto de Unity y nos pareció fácil e intuitivo de usar.

Figura 11. Visual Studio.

5.4. Base de datos

Para una completa interacción entre la aplicación y el paciente se realizó el diseño de una base de datos con el objetivo de guardar los datos adquiridos de cada uno de los pacientes para un posterior análisis.

Para el diseño e implementación de la base de datos implementada se ha escogido el servidor XAMPP, el cual es un sistema de gestión de base de datos MySQL, el servidor web Apache y las lenguas de interpretación PHP y Perl o gestión de aplicación.

Este servidor es de fácil acceso, y sus configuraciones son mínimas por lo cual para la persona que desea utilizar este tipo de servidor es de cómodo acceso y ayuda a optimizar tiempo del proceso.

Figura 12. Base de datos XAMPP.

5.5. Arduino UNO

Teniendo en cuenta, las especificaciones planteadas por el personal médico y los alcances establecidos en el presente proyecto, se empleó el Hardware Arduino uno, con el objetivo de analizar el comportamiento del paciente en un determinado ejercicio, por lo cual, se optó la presente plataforma debido a que ofrece diversos beneficios como:

- Bajo coste
- Fácil acceso e interacción con la plataforma (Simple y directo)
- Simplificación en el proceso de trabajar con microcontroladores.
- Multiplataforma, funciona con diferentes sistemas operativos.
- Software y Hardware ampliable y de código abierto.

Para dicha rutina se implementó una galga extensiométrica y se evaluó la interacción entre el paciente y el sensor, con el objetivo de medir la cantidad de veces que el paciente reproducía el ejercicio establecido.

Se empleó el Sensor de presión FSR406 debido a:

- El tamaño es acorde a las medidas de la rampa seleccionada.
- Excelente respuesta a la frecuencia.
- No intervienen campos magnéticos.
- Posee una compensación de temperatura.
- Alimentación con corriente alterna o continua.
- Medición de medidas dinámicas y estáticas.
- Correcta adhesión entre el sensor y el objeto de rehabilitación para correcta medida de deformación.

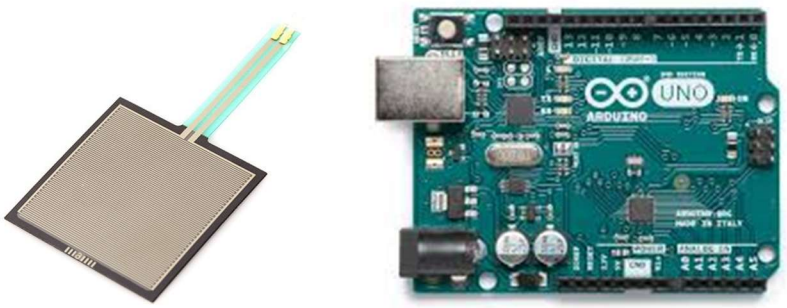


Figura 13. Sensor de presión FSR406 y Arduino UNO.

5.6. Rampa de Rehabilitación

Teniendo en cuenta una reunión organizada con el equipo de fisioterapeutas, se decidió añadir una rampa de rehabilitación para realizar el ejercicio propuesto por parte del personal médico. El ejercicio, que consta de pisar la rampa con un pie y volver a la posición inicial varias veces, es un

ejercicio básico en los tratamientos de los fisioterapeutas para aquellas personas que necesiten volver a caminar. Con el objetivo de analizar el progreso del paciente al entrar en contacto con la rampa de rehabilitación, se le añadieron las galgas de contacto.

Para el diseño de la rampa se hizo uso de la madera como material fundamental, debido a la facilidad para cortar, modificar y resistencia mecánica que este posee. Se le incorporo un sistema de bisagras para poder doblar las partes diseñadas y así convertirla en una rampa más cómoda para el usuario. consecutivamente, se pintó con una pintura antideslizante para evitar que el paciente se resbale en el momento que realiza el ejercicio.

Así mismo, se realizó el montaje y ensamblaje del circuito eléctrico diseñado en Arduino para un óptimo funcionamiento.

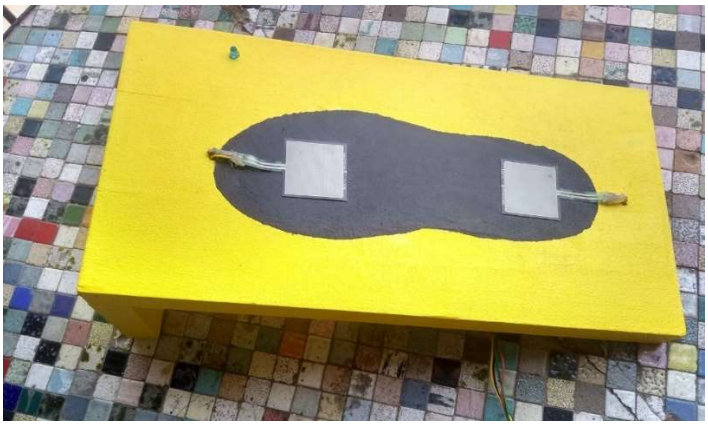


Figura 14. Rampa real

Las dimensiones utilizadas para la realización de la rampa fueron:

- 20 grados de inclinación respecto al suelo.
- 20 cm de ancho
- 40 cm de largo (Hipotenusa)

Diseñada para poder pisar con un pie y realizar el ejercicio. Así mismo, es un dispositivo manejable con el objetivo de poder ubicarla entre los apoyos (como las barras paralelas, o el caminador) que disponen en el centro para aquellas personas que no se puedan aguantar por sí mismas o que puedan perder el equilibrio.

Por otra parte, para el diseño de la rampa virtual, se seleccionó el programa SketchUp debido a que es un software de diseño de alta calidad para graficar y modelar objetos en tres dimensiones; así mismo, este programa es de fácil manejo para la exportación de .obj.

Figura 15. Programa de diseño 3D SketchUp.

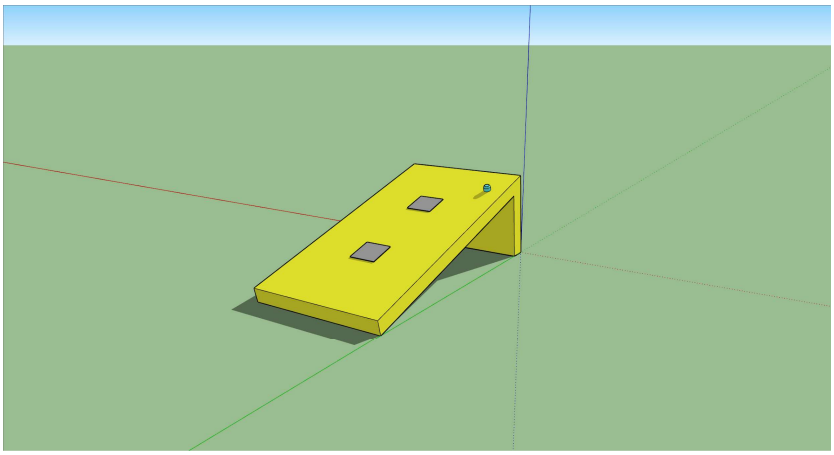


Figura 16. Rampa diseñada en SketchUp

6. DESARROLLO DEL PROYECTO

6.1. Características Específicas

Con el objetivo de cumplir con las metas trazadas por los fisioterapeutas y el cuerpo docente a cargo del presente proyecto, se diseñaron una serie de parámetros específicos que se tuvieron en consideración para realizar la aplicación deseada por el usuario y por el personal médico.

Parámetros en consideración:

- Fácil manejo por parte del personal y el paciente de forma intuitiva y concisa.
- Interactividad y comodidad de la aplicación teniendo en cuenta la selección de ejercicios especial para el paciente.
- Poder observar el movimiento a partir del avatar y poder compararlo con el paciente.
- Estudiar ángulos inferiores del paciente con el objetivo de compararlos y analizar la información suministrada por la plataforma (Ángulo de la cadera, rodilla y tobillo)
- Poder adquirir y guardar los datos de los pacientes para analizar y comparar posteriormente su rendimiento y progreso.
- Tiempo de duración de la repetición del ejercicio.

6.2. Ejercicios de rehabilitación evaluados

En el estudio de los miembros inferiores se tuvo en consideración:

- Comportamiento de las extremidades inferiores del paciente en interacción con una Rampa de rehabilitación.

Contexto: Ejercicio basado en la flexión de la cadera, rodilla y tobillo del paciente, teniendo en cuenta una elevación de 20 grados hasta que la extremidad inferior entre en contacto con el sensor ubicado en la rampa.

Parámetros analizados:

- Ángulo de flexión de la cadera - Costado derecho e izquierdo
- Ángulo de flexión de la rodilla - Costado derecho e izquierdo
- Ángulo de flexión-extensión del tobillo - Derecho e izquierdo
- Repeticiones del ejercicio
- Tiempo entre repeticiones (tiempo que tarda el paciente en volver a realizar el ejercicio)

6.3. Configuración del avatar en Unity

Para la correcta configuración del avatar se tuvo en cuenta los antecedentes del proyecto y las investigaciones previamente realizadas para comprender, continuar y culminar la aplicación de manera satisfactoria.

Para llevar a cabo este objetivo y entender el funcionamiento de todos los elementos que se integraron dentro de la aplicación es importante comprender la configuración y características estipuladas en el avatar y el entorno de Unity.

En primera instancia, se integró el avatar estudiado previamente en la escena principal en Unity, en donde el programador tiene libre albedrío de escoger, transformar y diseñar un entorno didáctico y ameno para la persona que se encuentra interactuando con la aplicación.

Se importó el avatar y sus texturas a la escena de estudio principal, teniendo en cuenta previamente la creación de un Gameobject a partir de un "T- Pose" y la opción "Humanoide" para la creación del modelo y sincronización con la posición predeterminada.

A continuación, a partir de la regla "T-Pose", se empalmaron todas las características de los huesos humanos con el avatar de estudio, en el cual se dividieron los GameObjects del avatar en diferentes GameObjects dependientes de este, para realizar un mapeo de las articulaciones del cuerpo humano y comprender el vínculo entre ellas.

Figura 17. Configuración del Avatar A) GameObjects de las articulaciones del avatar.

Para la sincronización de las articulaciones de referencia del avatar entre la plataforma Unity y el reconocimiento de las uniones en la herramienta Kinect fue importante realizar una comparación entre los diferentes puntos de medida y correlacionar todos los puntos del cuerpo humano, teniendo en cuenta así mismo la posición y distancia del paciente frente a la cámara Kinect. Los datos obtenidos de las uniones de las articulaciones fueron almacenados en la biblioteca de la Kinect a partir de Visual Studio.

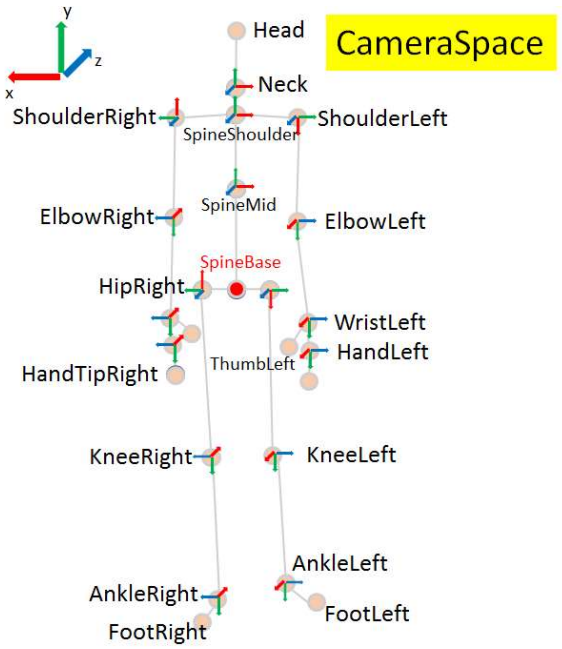


Figura 18. Configuración del Avatar. Puntos de referencia de las articulaciones del Sensor Kinect

Consecutivamente, con el objetivo de reproducir el movimiento conjunto de todas las articulares del avatar en tiempo real y obtener las dimensiones y puntos deseados se empleó las orientaciones a partir de un vector en tres dimensiones definiendo la postura absoluta. Dichas orientaciones se extrajeron teniendo en cuenta las rotaciones en función de su jerarquía y representación por cuaterniones, es decir, la descripción de la orientación en un espacio tridimensional.

6.4. Programación de la aplicación

6.4.1. Análisis de movimiento del Avatar

En este apartado, se evaluó y se observó el desarrollo y la comparación del movimiento implementado del avatar teniendo en cuenta los ejercicios requeridos por el personal médico y los resultados obtenidos, a partir de la implementación del lenguaje de programación C# teniendo en cuenta Visual Studio, el cual, hace referencia a un lenguaje orientado principalmente a objetos y componentes, fue diseñado por Microsoft como parte de su plataforma .NET, la cual tiene como objetivo la creación de aplicaciones sencillas.

Con el objetivo de reproducir los movimientos de los pacientes en tiempo real y estudiar el comportamiento del paciente se implementaron tres Scripts enlazados: "BodySourceManager", "Vector4" y "BodySourceView" [1].

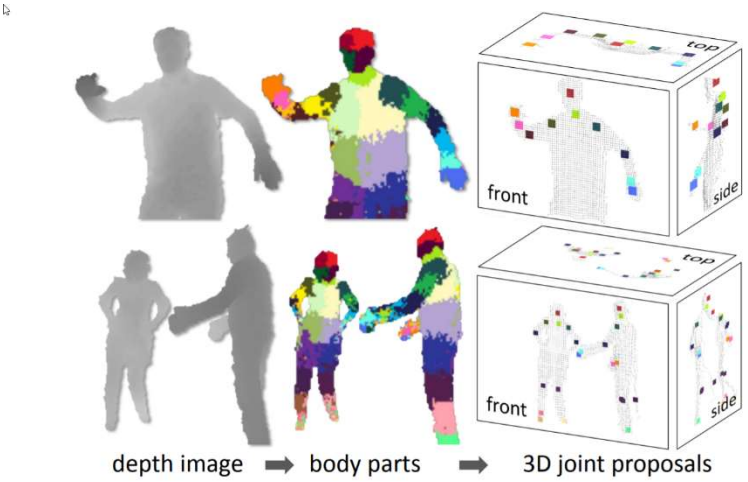


Figura 19. Captura de movimientos en tiempo real.

6.4.1.1. BodySourceManager:

Script imprescindible cuando se requiere trabajar con Kinect debido a que el programa permite que el usuario pueda tener una interacción correcta con la plataforma Unity y leer las articulaciones del cuerpo obteniendo la información escaneada en tiempo real. [Anexo B.1 BodySourceManager]

El presente Script proporciona tres comandos importantes: "UnityEngine","System.Collections" y "Windows.Kinect", los cuales son comandos utilizados para importar espacios de nombres, colecciones, clases y otro tipo de datos con el objetivo de categorizar las bibliotecas.

Así mismo, se contiene el espacio “System.Collections”, el cual contiene las interfaces y clases que definen las colecciones de los objetos (Listas, matrices de bits, tablas hash, diccionarios...). Por último, y no menos importante, se hace uso de "Windows.Kinect" el cual permite la interacción entre Unity y Kinect, con el objetivo de reconocer y leer las articulaciones del cuerpo.

6.4.1.2. Vector4:

El objetivo del Scripts Vector4, fue implementar un código en el cual las definiciones de los cuaterniones en las orientaciones de Unity y Kinect fuesen iguales; es decir, el vector de entrada fuese el vector quaternion obtenido en la Kinect teniendo en cuenta la inclinación del piso y el vector de salida fuese un nuevo vector de Cuaternion en Unity. [Anexo B.5 Vector4]

6.4.1.3. BodySourceView:

En el script BodySourceView se describieron todas las funciones implementadas para un correcto movimiento de avatar, estudiando, la orientación, rotación y dirección de cada uno de los puntos de referencia del avatar. [Anexo B.2 BodySourceView].

En primer lugar, se realizó la declaración de todas las variables y comandos a evaluar, en la cual se tuvo en cuenta los puntos de referencia de las articulaciones del cuerpo del avatar y la Kinect a partir de la obtención de datos de cada una de ellas y la importación de paquetes de Unity para Windows, con el objetivo de analizarlas posteriormente.

Seguidamente, se realizó la función de inicio a partir de la clase MonoBehaviour, la cual es la clase base de la que se deriva cada Script de Unity, este procedimiento se llevó a cabo a partir de la vinculación de las variables previamente declaradas a los puntos de referencia del avatar.

Así mismo, se realizó la función de actualización y se declaró las variables de rotación, con el objetivo de obtener los giros apropiados por cada una de las articulaciones. De igual modo, se tuvo en cuenta la inclinación del suelo, el movimiento natural del avatar y las jerarquías diseñadas anteriormente.

Consecutivamente, se realizó las funciones del seguimiento del Avatar y se determinaron las variables de cada Cuaternion, las cuales representan las rotaciones asignadas a las articulaciones del cuerpo.

Finalmente, teniendo en cuenta las propiedades de rotación, se realizó la asignación de cada una de las articulaciones con el objetivo de proporcionarle al avatar la dirección, rotación y movimiento de las articulaciones.

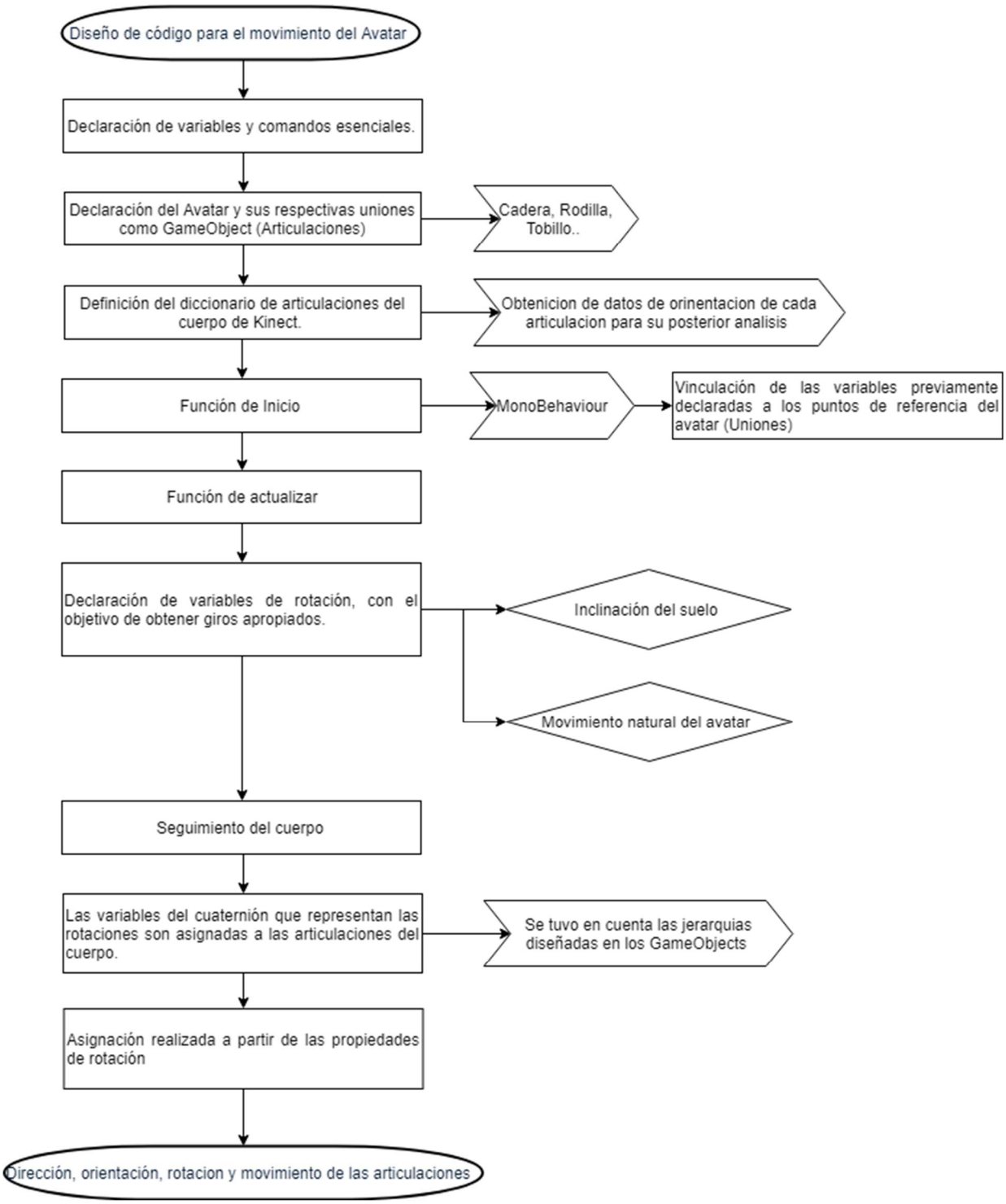


Figura 20. Configuración del Avatar. Diagrama de la orientación, rotación y dirección para el diseño de código del movimiento del avatar.(B.1 BodySourceManager)

A continuación, con el objetivo de gestionar la animación e incorporar todas las características del Avatar, se estudió el componente de PlayerController de Unity, el cual es el encargado de añadir y gestionar toda la animación del avatar; este componente contiene una máquina de estado capaz de coordinar la animación que se ejecuta en cada momento y la determinación del estado del movimiento deseado, continuando con el movimiento de interés.

Lo anteriormente mencionado, expresa, que cada uno de los personajes o avatares creados están vinculados directamente con diferentes acciones, los cuales dependerán del tipo de GamePlay. Estos estados y transiciones de un estado de máquina, pueden ser representados por un diagrama gráfico, donde los nodos representan los estados y los arcos representan las transiciones, como se puede observar en la Figura 16.

En el presente proyecto se tuvo en cuenta la animación de las extremidades inferiores como estado actual (Marcador), las cuales fueron integradas a la máquina de estado con el objetivo de animar está determinada parte del cuerpo para dar cumplimiento a el movimiento requerido por el fisioterapeuta y el paciente.

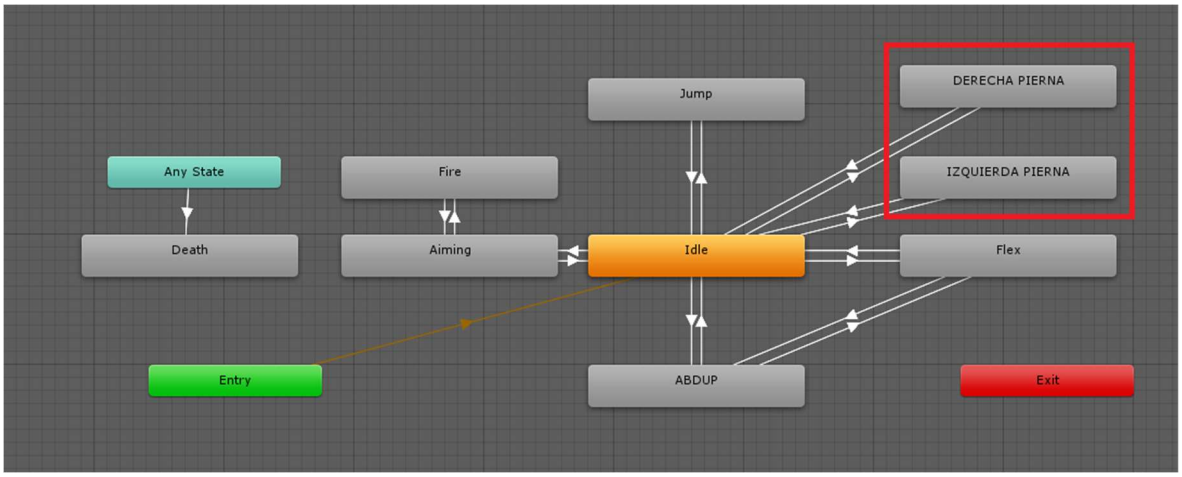


Figura 21. Diagrama de estados de animador.

6.4.3. Implementación del análisis de los movimientos del avatar

Una vez planteadas las especificaciones de análisis del avatar, se da inicio a la implementación, para ello se tiene en cuenta el Script "BodySourceView" el cual guarda la información de los movimientos que el usuario realiza durante la interacción con la plataforma. En este caso, los movimientos determinados por los miembros inferiores del paciente, evaluados a partir del estudio de sus ángulos.

- Evaluación de ejercicios de rehabilitación y ángulos de miembros inferiores.

A continuación, con el objetivo de evaluar la actividad del paciente a partir de los ejercicios de rehabilitación expuestos por el personal médico, se realizó el estudio de los ángulos de las extremidades inferiores; para lo cual, se tuvo en cuenta el estudio biomecánico de la cadera, rodillas y tobillos y las siguientes expresiones algebraicas para cumplir con los objetivos trazados.

Expresión Algebraica	**Ecuación**
Ecuación del plano general	$Ax + By + Cz + d = 0$
Vector entre dos puntos	$\vec{V} = (x2 - x1, y2 - y1, z2 - z1)$
Proyección entre un punto y un plano	$r = (x, y, z) + \lambda\,(v1, v2, v3)$
Ángulo entre dos vectores	$\cos(\alpha) = \frac{\vec{U} \cdot \vec{V}}{\lvert\vec{U}\rvert \cdot \lvert\vec{V}\rvert}$

Tabla 1. Expresiones algebraicas utilizadas para encontrar ángulos de extremidades inferiores.

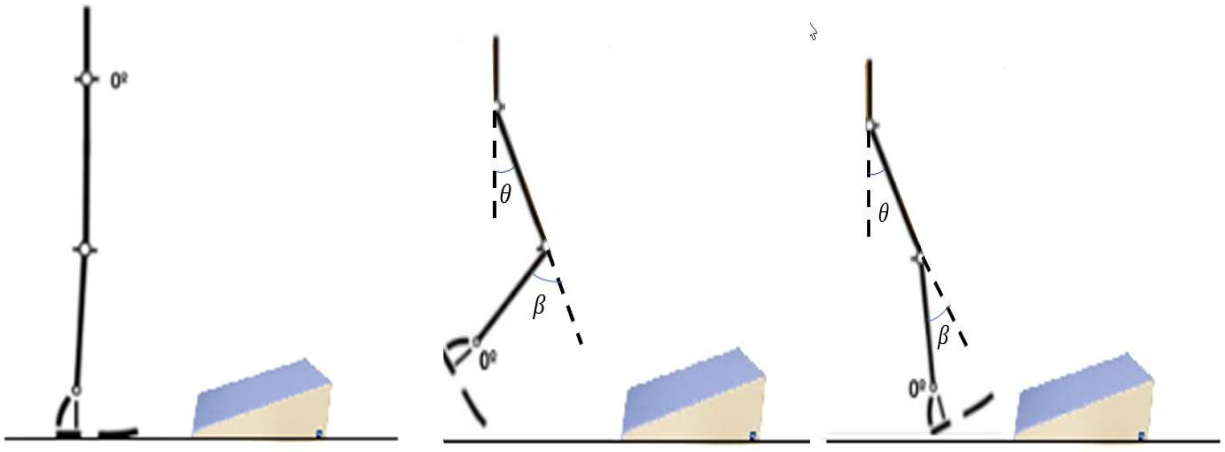


Figura 22. Ángulos inferiores estudiados cadera y rodilla.

Teniendo en cuenta las expresiones matemáticas expuestas anteriormente, se realizó el cálculo de los ángulos de flexión de la cadera y tobillo para los cuales fue necesario tener en cuenta la declaración de variables de cada uno de los puntos de referencia de las partes del cuerpo.

Para calcular los ángulos de flexión de la cadera se ejecutaron los siguientes pasos:

- Declaración de variables de la parte inferior del cuerpo.
- Cálculo del plano a través de la columna media y la cadera derecha.
- Se llama a la posición de la rodilla derecha.
- Proyección de la posición de la rodilla derecha al plano calculado.
- Se realiza un vector resultante entre la cadera y la posición proyectada entre los puntos de la columna.
- Se calcula el ángulo entre los vectores.
- Se realiza el mismo procedimiento para la evaluación de los ángulos entre la rodilla y cadera del costado izquierdo.

Para calcular los ángulos de flexión de la rodilla se ejecutaron los siguientes pasos:

- Declaración de variables de la parte inferior del cuerpo.
- Se realiza un vector resultante entre la posición de la cadera y de la rodilla.
- Se realiza un vector resultante entre la posición de la rodilla y el tobillo.
- Se calcula el ángulo entre los vectores.
- Se realiza el mismo procedimiento para la evaluación de los ángulos de la rodilla del costado izquierdo.

Para calcular los ángulos de flexo extensión del tobillo teniendo en cuenta la inclinación de la rampa de rehabilitación se ejecutaron los siguientes pasos:

- Declaración de variables de la parte inferior del cuerpo.
- Se realiza un vector resultante entre la posición del tobillo y el pie.
- Se realiza un vector resultante entre la posición de la rodilla y el pie.
- Se calcula el ángulo entre los vectores.
- Se realiza el mismo procedimiento para la evaluación de los ángulos del tobillo del costado izquierdo.

Figura 23. Implementación y configuración del avatar evaluado.

6.4.2. Rampa de rehabilitación

6.4.2.1. Resis_Sensor:

Script realizado con el objetivo de recibir, leer y almacenar los datos adquiridos por el Hardware Arduino, al estudiar el comportamiento del paciente en contacto con la rampa de rehabilitación. Así mismo, se evalúa y se cuenta el número de repeticiones y el tiempo en el que el paciente realiza el ejercicio propuesto a partir del uso del Sensor de presión FSR406, dicho sensor, posee una alta sensibilidad y precisión en el momento de adquirir los datos.

Para ellos fue necesario conocer y distinguir los puntos de presión en la planta del pie, con el propósito de conocer la correcta ubicación de los sensores en la rampa diseñada.

La planta del pie se puede dividir en 15 áreas específicas:

1. El talón (Áreas 1-3)
2. El arco (Áreas 4-5)
3. El metatarso (Áreas 6-10)
4. Los dedos (Áreas 11-15)

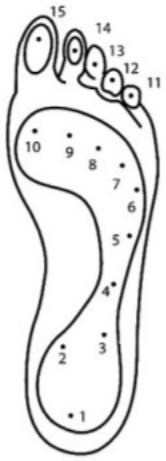


Figura 24. Puntos de presión en la planta del pie.

Las áreas mencionadas anteriormente, soportan la mayor parte del peso del cuerpo humano y ajustan el equilibrio del mismo, por lo cual, se seleccionaron dos partes de la planta del pie entre el talón y el metatarso, para situar los sensores debido a la precisión que poseen durante la actividad diaria de una persona.

El sensor de presión FSR406 seleccionado, se compone de un sistema extensiométrico se encuentran formados por:

- Elemento transmisor intermedio
- Amplificador
- Elemento registrador
- Elemento Procesador

El cual, mide la deformación de tracción y compresión al entrar en contacto con la planta del pie del paciente, debido a que convierte la fuerza, peso, presión, tensión u otra magnitud física, en un cambio de resistencia eléctrica.

El sensor FSR406 fue conectado a un circuito eléctrico, el cual proporciona la obtención de señal eléctrica enviada desde las galgas extensiométricas hasta el Arduino, con el objetivo de adquirir los datos correspondientes a dicha deformación y procesarlos posteriormente. [Anexo B.8 Galgas]

El circuito implementado (Figura No.20) se incorporó a la rampa de rehabilitación física descrita en el apartado 5.6, para un correcto funcionamiento e interacción con el paciente en el momento de realizar el ejercicio de rehabilitación establecido.

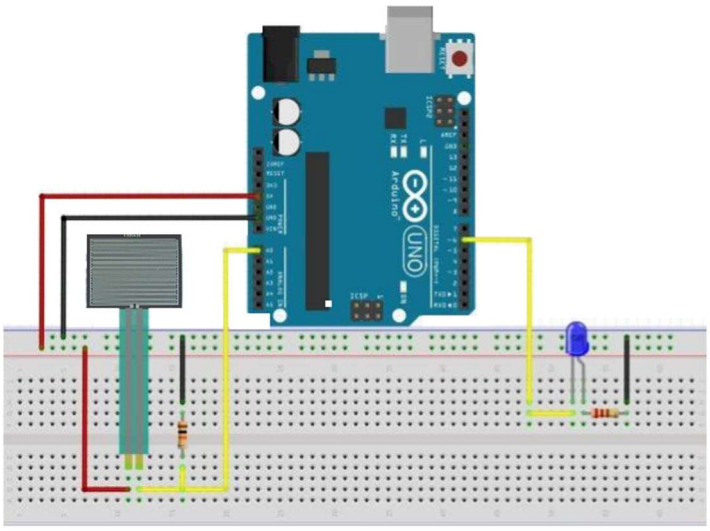


Figura 25. Circuito implementado (Conexión Arduino- Galga extensiométrica).

Consecutivamente, se realizó la conexión entre Unity y el Hardware Arduino, en la cual se leyeron los datos y se almacenaron los datos en un array con el objetivo de mostrarlos en una escena específica en Unity.

6.4.3. Creación de Escenarios 3D

Teniendo en cuenta los objetivos de la aplicación médica diseñada y a las personas a las cuales va dirigido el presente proyecto de investigación, se realiza un escenario de ambiente realista y dinámico, para ello se hizo uso del Asset Store de Unity, con el objetivo de elaborar diferentes escenas ambientadas.

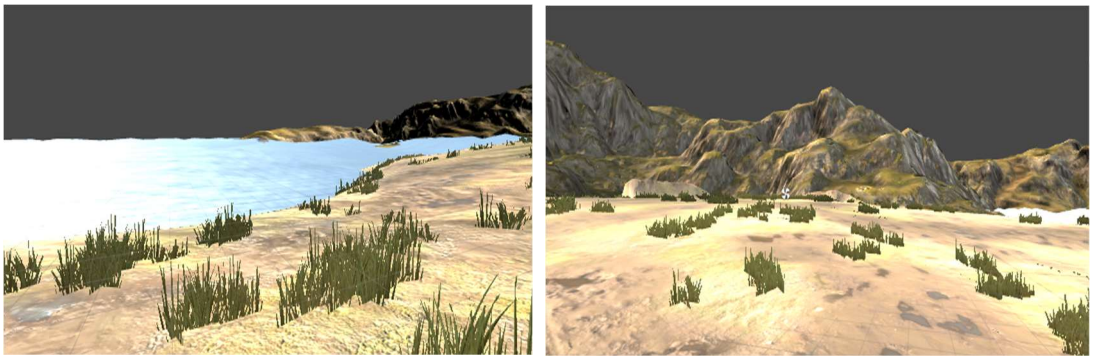


Figura 26. Ambiente de los escenarios.

La primera de ellas escenifica un panel de control de los datos de los pacientes por parte del personal médico. El segundo es un panel donde el usuario puede seleccionar la rutina de ejercicios que desea realizar y, por último, el tercer escenario, es donde el usuario puede sumergirse e interactuar con todos los elementos virtuales de la aplicación específicamente para el ejercicio de interés.

6.4.3.1. Escenario de registro de datos

El escenario para el registro de datos, se diseñó con el objetivo de facilitar el ingreso de los médicos y de los pacientes en el programa realizado con el objetivo de agilizar el proceso.

Para cumplir con este objetivo, se diseñó una escena de “inicio de sesión” (Figura 21); en la cual, el fisioterapeuta y el paciente introducen los datos personales para dar inicio a el programa de rehabilitación.

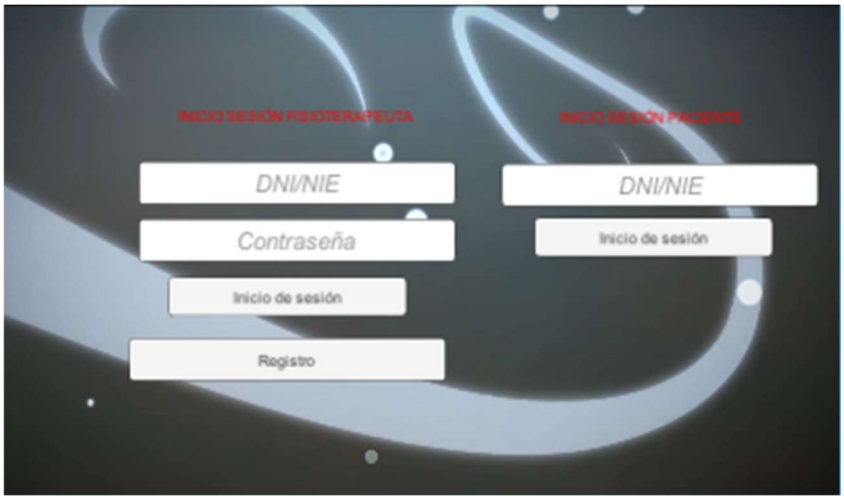


Figura 27. Escena de inicio de sesión.

Al estar en contacto por primera vez la aplicación con el paciente, se diseñó una escena para registrar a el paciente que se desea analizar; en la cual, se deben proporcionar datos como, nombre, apellido, DNI/NIE y una nueva contraseña.

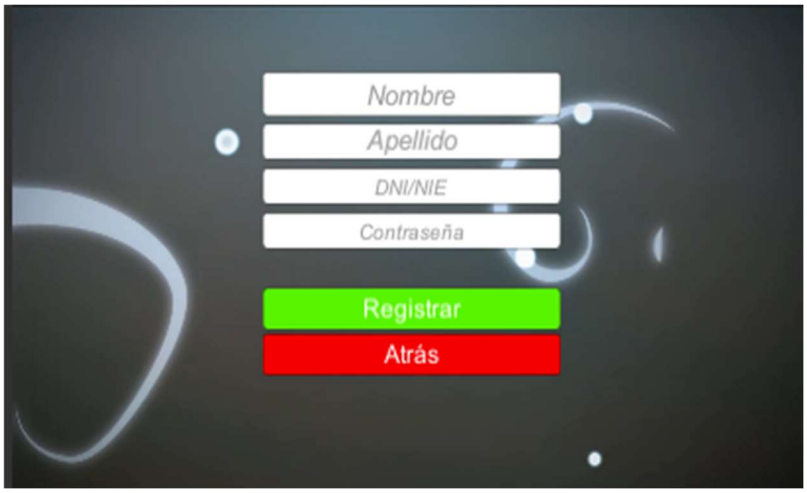


Figura 28. Escena de registro.

6.4.3.2. Escenario de Selección del ejercicio.

Al obtener la información suministrada por el paciente y los datos del cuerpo médico, se procedió a realizar el escenario de selección del ejercicio.

Para cumplir con este propósito, se realizó el diseño y programación de un canvas; en el cual, se adjuntaron todos los ejercicios propuestos por los especialistas, dividiéndolos según la rutina de

rehabilitación que el paciente necesitará realizar teniendo en cuenta una sección especial para el Cuerpo Superior y otra sección para el Cuerpo Inferior, siendo nuestro objeto de estudio, el cuerpo inferior de la rutina de rehabilitación.

Para el cumplimiento de lo anteriormente descrito, el primer canvas realizado tuvo como objetivo indicar y visualizar todos los ejercicios estudiados en el presente proyecto teniendo en cuenta diferentes ButtonClicks, con el propósito de obtener una interacción entre diferentes escenarios.

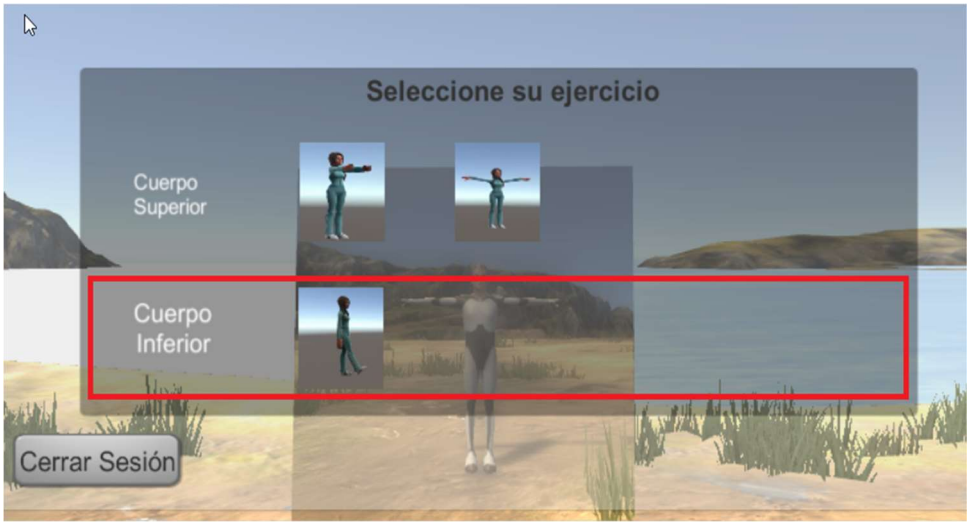
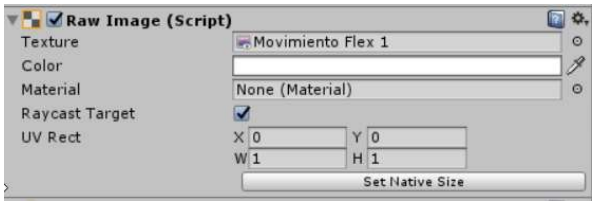


Figura 29. Escenario de selección de ejercicio.

Para obtener el resultado esperado se empleó una serie de instrucciones y acciones dirigidas a el botón del movimiento seleccionado, con el objetivo de cumplir diferentes funcionalidades dentro de la aplicación. Así mismo, se vinculó un video corto interactivo del ejercicio de Flexión de la cadera, rodilla y tobillo teniendo en cuenta un grado de inclinación específico, con el objetivo de identificar más fácilmente el ejercicio requerido por el paciente.



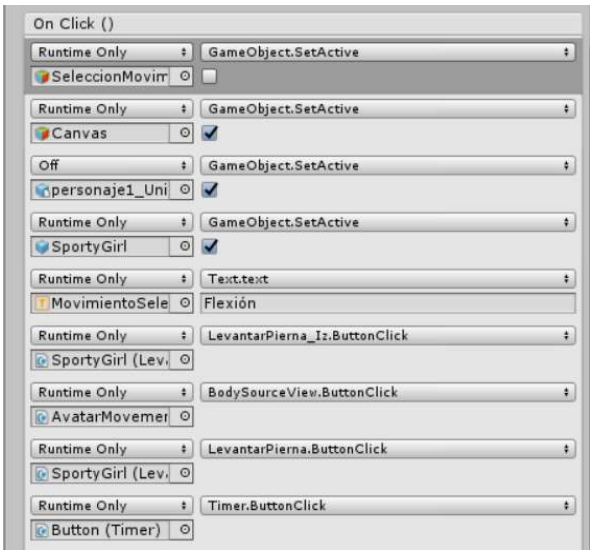


Figura 30. Instrucciones realizadas por el ButtonClick de miembro inferior.

6.4.3.3. Escenario de Flexión de miembros inferiores.

Consecutivamente, luego de seleccionar el movimiento inferior que se desea, se incorporó un nuevo escenario con el objetivo de observar la actividad del paciente en interacción con el avatar y la plataforma virtual.

Para ello se diseñaron las siguientes herramientas interactivas:

1. **SportyGirl:** Modelo programado con el objetivo de realizar el movimiento del ejercicio correctamente con el propósito de que paciente comprenda el ejercicio propuesto y consecutivamente lo imite y reproduzca de igual forma.

   Por este motivo, se implementó, el Script “Levantar Pierna” y “Levantar Pierna_Iz”, que como su nombre indica es un Script formado con el objetivo de que el modelo programado, levante las extremidades inferiores en un tiempo determinado a un concreto ángulo de inclinación. [Anexo B.6 LevantarPierna y B.7 LevantarPierna_Iz]

   Así mismo se adiciono, un Script llamado “Timer”, el cual, se encuentra conectado con la herramienta interactiva (4), con el objetivo de conocer el tiempo estimado que el paciente necesita comenzar a realizar la actividad contemplada. [B.4 Timer 7]

2. **Avatar:** Modelo animado, el cual tiene como objetivo la imitación de los movimientos del paciente; el avatar, contiene todas las características fisiológicas del cuerpo humano y la conexión con todas herramientas diseñadas para un óptimo funcionamiento como; la rampa de rehabilitación, el semáforo de inicio y los ángulos analizados; las cuales se encuentran descritas y programadas en el Script "Body Source View"
3. **Semáforo y texto descriptivo:** Dispositivo de señalización luminosa, el cual consta de tres luces (Roja, amarilla y verde) el cual dará la indicación inmediatamente luego de terminado el ejercicio por el modelo programado (a) para que el paciente continúe la rutina; así mismo, se realizan las indicaciones de forma escrita programado simultáneamente con el semáforo a partir de las siguientes frases:
    - Observe el ejercicio que debe realizar (Rojo)
    - ¡Ahora es tu turno! (Verde)
4. **Timer:** Contador inverso de los segundos restantes para empezar a realizar la rutina por parte del paciente, el código diseñado para realizar esta acción se encuentra en el Script "Contador_1" [Anexo B.4 Timer]
5. **Espejo:** Superficie que refleja los movimientos del paciente en la consola a través del avatar, con el objetivo de que el paciente observe y compare sus movimientos con los movimientos del modelo programado.
6. **Ángulos medidos:** Panel de control realizado con el objetivo de indicar los ángulos de flexión de la cadera, rodilla y tobillo máximos y actuales de los dos costados del cuerpo del paciente cuando este realiza el ejercicio propuesto; a partir de la interacción de controles interactivos. Para realizar dicha acción se hizo uso del Script "Body Source View" debido a que permite la unión entre la plataforma y las extremidades del paciente.
7. **Botones de repetición:** también llamados, 'Inicio de movimiento de flexión izquierda y derecha', los cuales repite el ejercicio del costado que se desea realizar cuando el cuerpo médico desea observar con detalle el ángulo de un determinado movimiento. Cada uno de los botones se encuentra enlazado con los Scripts "Levantar Pierna" y "Levantar Pierna_Iz".
8. **Rampa / Flexión:** Con el objetivo de que el paciente tenga una mejor experiencia al entrar en contacto con la aplicación, se realizó una rampa virtual, la cual al ponerla en contacto con el paciente se localizara y sincronizará con el avatar en tiempo real, lo anteriormente descrito se realizó a partir del Script "Resis_Sensor".
9. **Número de repeticiones:** Contador de veces que el usuario realiza el ejercicio al ponerse en contacto con la rampa de rehabilitación teniendo en cuenta la herramienta interactiva (8).
10. **Tiempo entre repeticiones**: evaluación del intervalo del tiempo que el paciente necesita para volver a realizar la siguiente repetición; es decir, el tiempo desde que el paciente inicia el movimiento y hace contacto con el sensor ubicado en la rampa de rehabilitación.

11. **Botón atrás:** Botón diseñado con el objetivo de retroceder y observar todos los ejercicios de rehabilitación propuestos en la plataforma para comenzar una nueva rutina.

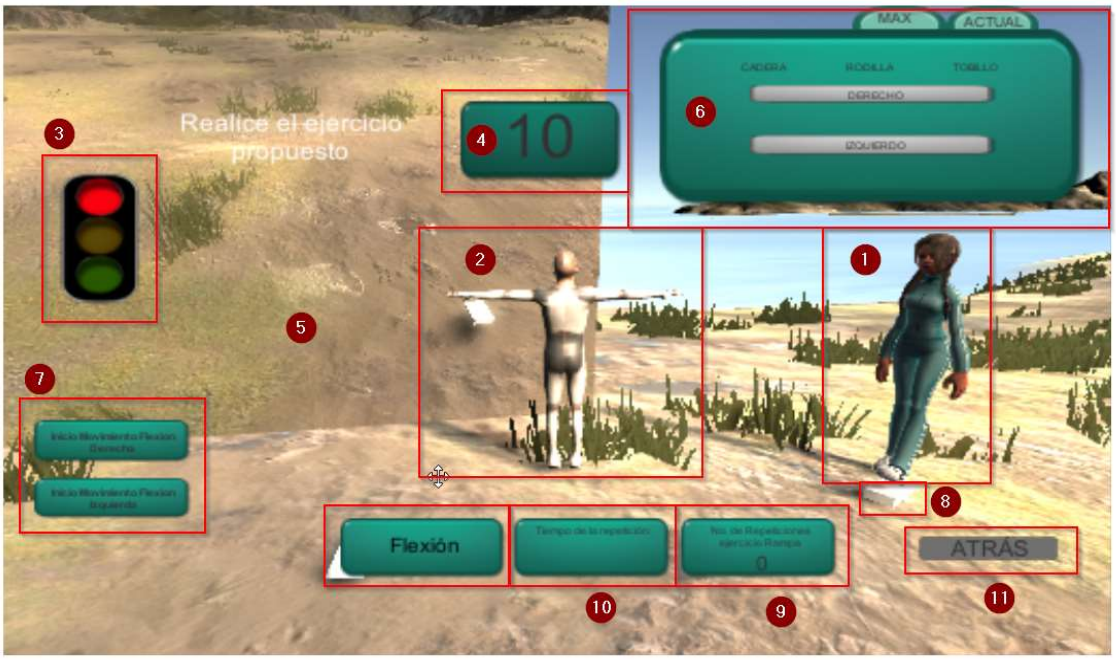


Figura 31. Canvas de cada uno de los datos medidos en el ejercicio propuesto.

7. RESULTADOS Y ANALISIS DE LA APLICACIÓN

Partiendo de los objetivos trazados en la etapa inicial del proyecto de grado, su desarrollo y el producto final obtenido, se precisan los siguientes resultados tanto a nivel experimental, tecnológico y personal.

Para poder comprobar el funcionamiento de nuestro trabajo y obtener resultados satisfactorios y verídicos, nos dirigimos a la ADFO (Associació de Disminuïts Físics d'Osona) todo el equipo de trabajo y cinco pacientes, los cuales realizaron los ejercicios propuestos en la aplicación medica diseñada.

Los pacientes llegaron a la sala condicionada anteriormente para el estudio, en la cual se realizó un itinerario del día, con el objetivo de que la duración de la espera del paciente fuese lo más amena y corta posible, así mismo se realizó una explicación a cada uno de los voluntarios de los ejercicios con los cuales iban a interactuar.

Hora	**Pacientes**	**Ejercicio de rehabilitación**
9:00	Paciente 1	Rampa
9:30		Rampa +VR
10:00	Paciente 2	Rampa
10:30		Rampa +VR
11:00	Paciente 3	Rampa
11:30		Rampa +VR
12:00	Paciente 4	Rampa
12:30		Rampa +VR
13:00	Paciente 5	Rampa
13:30		Rampa +VR

Tabla 2. Cronograma de Pruebas- Pacientes

A los primeros 3 pacientes se les explicó verbalmente la dinámica para la óptima realización del ejercicio, mientras que a los últimos dos se les sugirió que detallaran el ejercicio en la pantalla a partir del avatar SportyGirl, el cual explicaría como debían realizar el ejercicio correctamente teniendo en cuenta diez repeticiones por ejercicio diseñado con el objetivo de comprobar los ángulos adquiridos por la Kinect.

Consecutivamente, se les realizo un cuestionario de satisfacción con el objetivo de conocer su opinión frente a la aplicación diseñada (Tabla 3. Valoración de la aplicación

Y así mismo, realizar mejoras de la aplicación en un futuro. Algunos de los pacientes tenían dificultad para hablar, por este motivo se seleccionó un método fácil de respuesta, las cuales eran de sí/no.

Paciente	**Sí**	**No**	**Ns/Nc**
Ha comprendido el funcionamiento de la aplicación fácilmente?			
Cree que le resultaría más entretenido realizar los ejercicios con la aplicación que sin ella?			
Podría realizar los ejercicios sin ayuda sin las gafas VR puestas?			
Podría realizar los ejercicios sin ayuda con las gafas VR puestas?			
Ha sentido alguna molestia como mareos, o vértigo, u otro con las gafas VR?			
Prefiere realizar el ejercicio con las gafas VR?			
Que mejoras cree que serían convenientes para la aplicación en un futuro?			

Tabla 3. Valoración de la aplicación

Teniendo en cuenta los resultados proporcionados por los pacientes se realizó un análisis estadístico con los resultados obtenidos.

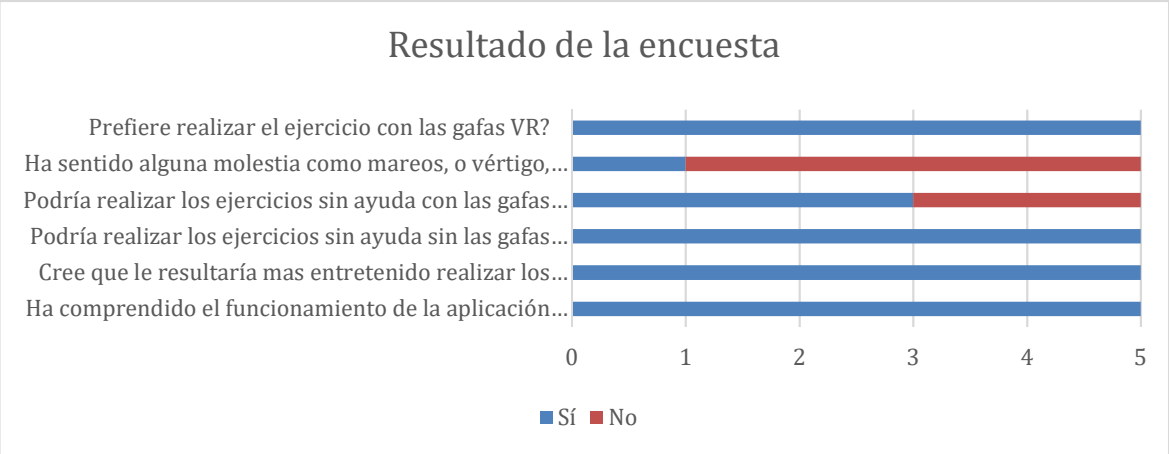


Figura 32. Gráfica de los resultados de las encuestas.

Se puede evidenciar que los resultados de la encuesta realizada son positivos y el 100% de los pacientes estudiados aseguraron preferir realizar la rutina de rehabilitación con la aplicación médica y Realidad Virtual.

Cabe destacar, que algunos de los individuos que han sufrido ictus presentaban cierta dificultad para colocarse las gafas solos o para posicionarse delante de la rampa. Es relevante dar a conocer que un paciente nos comentó que sufrió algún mareo en el momento de quitarse las gafas.

Figura 33. Paciente realizando el ejercicio con la ayuda de la fisioterapeuta.

Figura 34. Paciente realizando el ejercicio sin ayuda y sin gafas VR

Observamos que los pacientes se encontraban muy cómodos con la aplicación, con una actitud muy optimista e ilusionados. Creemos que, la mejora del estado anímico del paciente, que supone la realización del ejercicio con las gafas es un importante argumento para seguir avanzando con la utilización y mejora de la aplicación, debido a que incrementa el interés y la motivación en el momento de realizar el proceso de rehabilitación siendo eficaz el resultado.

Al finalizar con los pacientes, nos reunimos con las fisioterapeutas de ADFO y comentemos los aspectos positivos de la aplicación y qué se podría mejorar en un futuro. El tamaño reducido de la rampa y la Kinect permitían que los pacientes pudieran disponer de un punto de apoyo como puede ser un bastón o un caminador puesto del revés.

No obstante, en el caso de que el paciente precisara ayuda por el personal, cuando un profesional se aproximaba, la Kinect detectaba las dos personas y los resultados no eran correctos. Otro problema a resolver con la obtención de los resultados fue en la precisión de los ángulos del

tobillo, debido a que la Kinect tiene problemas para detectar el pie como podemos observar en la Figura 35. Visión de la Kinect con la cámara térmica.

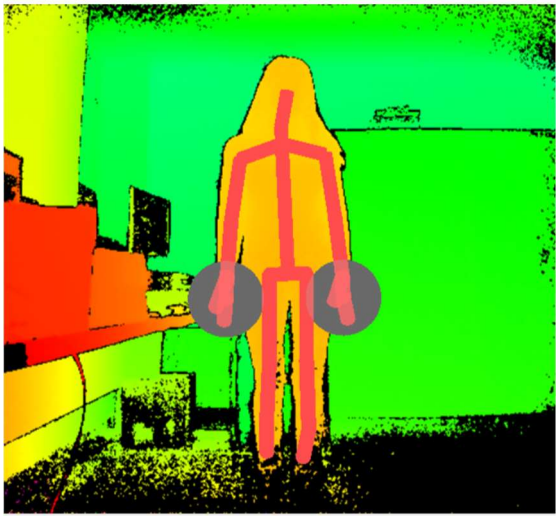


Figura 35. Visión de la Kinect con la cámara térmica.

Respeto los resultados personales, en primer lugar, se destaca el aprendizaje obtenido en el software Unity, teniendo en cuenta los conocimientos teóricos y prácticos dentro de esta área; debido a que fue una de las herramientas fundamentales para el diseño de la aplicación de rehabilitación, los conocimientos adquiridos en el lenguaje de programación C#, el diseño de interfaces y la implementación y gestión de bases de datos.

Así mismo, comparando los objetivos trazados en un inicio del proyecto y la recepción obtenida por parte del cuerpo médico, rehabilitador y del paciente al entrar en contacto con ella, se determina que la aplicación realizada es una herramienta óptima en los procesos de rehabilitación tanto a nivel de mejora de la capacidad motriz como a nivel de mejora del estado de ánimo y social.

8. ANÁLISIS DEL IMPACTO AMBIENTAL

El proyecto VIRTUAL MIRROR es un proyecto de investigación el cual no tiene un impacto ambiental directo. Sin embargo, para el correcto funcionamiento y desarrollo del proyecto, la aplicación genera un consumo de energía eléctrica debido al uso de los dispositivos electrónicos, como, el ordenador, la Kinect y el Arduino uno; así mismo, se considera que los componentes electrónicos que cada uno de estos dispositivos poseen (resistencias, cámaras, placas, transistores, entre otros) son producidos por materias primas que han tenido una producción y preparación previa, en los cuales se hace uso de diferentes contaminantes para la fabricación de estas piezas.

Es importante tener en cuenta que el componente al finalizar su vida útil, debe tener una adecuada gestión de desechos electrónicos, a partir de la normativa para la protección ambiental la cual exige y describe el uso, producción y distribución de artefactos tecnológicos de una manera responsable y la Directiva 2012/19/UE del parlamento Europeo y del Consejo, acerca de los residuos de equipos eléctricos y electrónicos - RAEE, la cual establece la recogida, reciclaje y correcta recuperación de estos [5][10].

Así mismo, al implementar la madera como material principal en el diseño de la rampa de rehabilitación, se puede estipular que al ser un material biodegradable, natural, renovable y reciclable no contamina, debido a que no ha estado en contacto con ningún proceso industrial o químico y teniendo en cuenta la pintura utilizada, esta tampoco representa un impacto ambiental mayor debido a que se seleccionó una pintura ecológica.

9. ANÁLISIS ECONÓMICO

Para el estudio económico del presente proyecto, se consideraron diversos factores, que se describirán a continuación:

9.1. Costo del Personal

El actual proyecto tuvo la participación de dos Ingenieros para el diseño de la aplicación; los cuales iniciaron el día 3 de enero del 2019 y finalizaron el día 3 de junio del mismo año. Teniendo en cuenta 25 horas por ECTS Y sabiendo que el proyecto final de grado son 24 ETCS, se dedicó 600 h/ ingeniero para la realización del presente trabajo.

Rol	Nº de personal	Tiempo (h)	Coste por hora (€)	Coste (€)
Ingeniero Biomédico	1	600	30,00	18000,00
Ingeniero Mecánico	1	600	30,00	18000,00
Coste Total de Personal				36000,00

Tabla 4. Coste del Personal

Así mismo, se tuvo en cuenta el coste de la seguridad social la cual equivale al 23,6 % de la base de la cotización, la cual, para los dos ingenieros el coste fue:

Rol	Coste seguridad Social
Ingeniero Biomédico	3398,4
Ingeniero Mecánico	3398,4
Coste total de la Seguridad Social del Personal	6796,8

Tabla 5. Coste de Seguridad Social del personal.

9.2. Equipamiento

9.2.1. Costo del Hardware

En el presente apartado se describe el Hardware implementado durante la realización del proyecto.

Producto	Precio por Unidad (€/u)	Unidades (€)	Vida útil estimada (meses)	Tiempo de uso (meses)	Precio residual (€/u)	Amortización (%)	Total (€)
Kinect v2 Sensor	85,00	1,00	36,00	4,00	40,00	1,30	5,00
LAM Computer	1500,00	1,00	48,00	4,00	350,00	24,00	95,80
HTC Vive	600,00	1,00	36,00	4,00	200,00	11,10	44,40
Windows Adapter	40,00	1,00	36,00	4,00	10,00	0,80	3,30
Coste total del Hardware							148,60

Tabla 6. Coste de Hardware Implementado en la aplicación.

9.2.2. Costo del Software

En el presente apartado se describe el Software implementado durante el diseño de la aplicación médica.

Producto	Precio por Unidad (€/u)	Unidades (€)	Total (€)
Curso Formación Unity	9,00	1	9,00
Microsoft Kinect for Windows SDK	0,00	1	0,00
Microsoft Visual Studio	0,00	1	0,00
Steam VR	0,00	1	0,00
Unity version 5.6.0f3	0,00	1	0,00
Coste total del Software	0,00	1	9,00

Tabla 7. Coste de Software Implementado en la aplicación.

9.2.3. Costo de servicios

En el presente apartado se describe el costo de los servicios implementados durante el proceso de realización de la aplicación médica.

Servicio	Precio por mes (€/u)	N.º de Meses	Total (€)
Electricidad	30,00	4	1500,00
Internet	48,00	4	192,00
Coste total de servicios			1692,00

Tabla 8. Coste de los servicios utilizados.

9.2.4. Costo Totales

En el presente apartado se describe el costo total de la aplicación médica.

Costos totales	Precio por mes (€/u)
Coste Total del Personal	36000
Coste total de la Seguridad Social del Personal	8496
Coste total del Hardware	148,6
Coste total del Software	9
Coste total de los servicios	1692
Costo total	**46345,6111**

Tabla 9. Coste total de la aplicación

10. PROYECCIÓN DE LA INVESTIGACIÓN

El presente proyecto de investigación tiene un elevado potencial en el campo de la medicina, debido a su método interactivo y efectivo, siendo agradable para cualquier tratamiento físico que se pueda estudiar a partir de la interacción de este tipo de softwares.

Quedar finalistas en las becas Carles Capdevila demuestra que este proyecto tiene potencial. Las becas Carles Capdevila son creadas por el diario 'Ara' y 'La fundació Bancària La Caixa' para premiar proyectos comprometidos con la educación y la cura de las personas [3].

En futuro se puede realizar el estudio e integración de ejercicios de rehabilitación en diferentes partes del cuerpo y el registro gráfico de la progresión del paciente, mostrando el avance obtenido durante la interacción con la plataforma. De igual manera, la implementación de una plataforma con niveles de complejidad cuando el paciente se encuentra en un proceso autónomo o de larga duración de su rehabilitación podría ser una excelente manera de estudiar todo su proceso para futuras investigaciones por parte del personal médico.

Así mismo, utilizando algoritmo Dynamic Time Warping, se puede realizar la verificación y comparación de un registro correcto guardado en la base de datos y el movimiento del paciente que se desea analizar. Esto se podría llevar acabo, a partir de la detención e indicación de un cuadro de texto si el paciente realizará un movimiento erróneo. En este instante la articulación o parte del cuerpo estudiada cambiaria de tonalidad, en consecuencia, del error realizado, teniendo en este momento una justificación del error cometido y la correcta postura para continuar el ejercicio de rehabilitación.

11. CONCLUSIONES

Actualmente, la realidad virtual está creciendo exponencialmente y cada vez se están realizando nuevas investigaciones y adaptaciones más arraigadas a las necesidades diarias de las personas, lo cual permite una excelente aceptación por parte de todos los usuarios que interactúan con este tipo de plataformas. Así mismo, día a día y gracias a los avances tecnológicos, la demanda y nuevas implementaciones entre hardware y softwares la población pueda acceder a este tipo de tecnología con mayor facilidad y bajo costo.

Al observar la interacción entre el personal médico, el paciente y la aplicación médica diseñada, se pudo evidenciar que es una herramienta intuitiva, amigable, innovadora y de fácil acceso, totalmente funcional para cumplir con los objetivos trazados a lo largo del proyecto por parte de todo el grupo de trabajo al realizar las pruebas piloto con la aplicación medica diseñada. (Versión beta)

Así mismo, teniendo en cuenta las pruebas realizadas y análisis de los datos obtenidos del paciente a través de la aplicación, se puede observar que los valores estudiados se encuentran dentro de un margen de error aceptable, siendo confiable y precisa la aplicación para el estudio y el seguimiento de los procesos de rehabilitación de cada uno de los pacientes que necesiten este tipo de rutinas.

Una de las mayores dificultades del presente proyecto ha sido la implementación y localización de objetos reales e inmersos sincronizados en tiempo real, en este caso, la rampa de rehabilitación, debido a la oclusión en algunas partes de cuerpo como el pie y el tobillo del paciente teniendo en cuenta la exactitud de los puntos tomados por la Kinect al entrar en contacto con la Rampa y movimientos complejos que el sistema no puede percibir.

De igual manera, se recomienda el uso de un sistema de versiones Git, para el desarrollo de futuros proyectos corporativos como GitHub.

A nivel personal, se han logrado todos los objetivos y expectativas que se establecieron en el inicio del proyecto, debido a que, se adquirió y reforzó conocimientos en el área de la programación, diseño de interfaz, trabajo grupal, adquisición de datos, entre otras múltiples herramientas utilizadas para culminar satisfactoriamente el proyecto. Así mismo, nos encontramos satisfechos con el proyecto de grado realizado debido a la integración conceptos fundamentales vistos en nuestra formación como ingenieros logrando ser un proyecto compuesto por componentes acordes a nuestras áreas de interés y líneas de investigación futuras.

12. BIBLIOGRAFIA

[1] Archiveddocs. «System Requirements». Accedido 24 de mayo de 2019. https://docs.microsoft.com/en-us/previous-versions/aa700918(v%3dmsdn.10).

[2] «Arduino - Home». Accedido 3 de junio de 2019. https://www.arduino.cc/.

[3] «Beca Carles Capdevila 2019 - Innovar per educar». Ara.cat. Accedido 24 de mayo de 2019. https://becacarlescapdevila.ara.cat/.

[4] «capitulo2.pdf». Accedido 31 de mayo de 2019. http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/hernandez_s_f/capitulo2.pdf.

[5] Directive 2012/19/EU of the European Parliament and of the Council of 4 July 2012 on waste electrical and electronic equipment (WEEE) Text with EEA relevance, Pub. L. No. 32012L0019, OJ L 197 (2012). http://data.europa.eu/eli/dir/2012/19/oj/eng.

[6] «Elementos de la realidad aumentada – Realidad Aumentada – Augmented Reality». Accedido 3 de junio de 2019. http://www.realidad-aumentada.eu/es/elementos-de-la-realidad-aumentada/.

[7] El-Shamy, S., y R. Alsharif. «Effect of Virtual Reality versus Conventional Physiotherapy on Upper Extremity Function in Children with Obstetric Brachial Plexus Injury». *Journal of Musculoskeletal & Neuronal Interactions* 17, n.º 4 (01 de 2017): 319-26.

[8] Gaggioli, Andrea, Emily A. Keshner, y Patrice L. Weiss. *Advanced Technologies in Rehabilitation: Empowering Cognitive, Physical, Social and Communicative Skills Through Virtual Reality, Robots, Wearable Systems and Brain-Computer Interfaces*. IOS Press, 2009.

[9] García-Rudolph, Alejandro, David Sánchez-Pinsach, Eloy Opisso Salleras, y Josep María Tormos. «Subacute Stroke Physical Rehabilitation Evidence in Activities of Daily Living Outcomes: A Systematic Review of Meta-Analyses of Randomized Controlled Trials». *Medicine* 98, n.º 8 (febrero de 2019): e14501. https://doi.org/10.1097/MD.0000000000014501.

[10] Hidalgo Aguilera, Luis. «La basura electrónica y la contaminación ambiental». *Enfoque UTE* 1, n.º 1 (31 de diciembre de 2010): 46. https://doi.org/10.29019/enfoqueute.v1n1.16.

[11] Kefaliakos, Antonios, Ioannis Pliakos, Panagiotis Kiekkas, Martha Charalampidou, y Marianna Diomidous. «Virtual Reality in the Rehabilitation of Patients with Neurological Disorders». *Studies in Health Technology and Informatics* 226 (2016): 45-47.

[12] Lange, Belinda, Stacey George, Judith E Deutsch, Gustavo Saposnik, Maria Crotty, y Kate E Laver. «Virtual reality for stroke rehabilitation». *The Cochrane Database of Systematic Reviews* 2017, n.º 11 (20 de noviembre de 2017). https://doi.org/10.1002/14651858.CD008349.pub4.

[13] Maturana, Jesús. «HTC Vive, análisis: esto sí que es realidad virtual interactiva». Xataka, 13 de mayo de 2016. https://www.xataka.com/analisis/htc-vive-analisis-esto-si-que-es-realidad-virtual-interactiva.

[14] «MECANISMOS BÁSICOS DE REALIDAD VIRTUAL». Accedido 3 de junio de 2019. http://www.difementes.com/realidadvirtual/clasificacionsegunlainterfaz.html.

[15] Palma, Gisele Carla Dos Santos, Tatiana Beline Freitas, Giordano Márcio Gatinho Bonuzzi, Marcos Antonio Arlindo Soares, Paulo Henrique Wong Leite, Natália Araújo Mazzini, Murilo Ruas Groschitz Almeida, José Eduardo Pompeu, y Camila Torriani-Pasin. «Effects of Virtual Reality for Stroke Individuals Based on the International Classification of Functioning and Health: A Systematic Review». *Topics in Stroke Rehabilitation* 24, n.º 4 (2017): 269-78. https://doi.org/10.1080/10749357.2016.1250373.

[16] «Realidad Virtual aplicada a la salud - Usos para pacientes y médicos». *Innoarea Projects* (blog), 13 de enero de 2017. http://www.innoarea.com/realidad-virtual-aplicada-a-la-salud/.

[17] Rogers, Jeffrey M., Jonathan Duckworth, Sandy Middleton, Bert Steenbergen, y Peter H. Wilson. «Elements Virtual Rehabilitation Improves Motor, Cognitive, and Functional Outcomes in Adult Stroke: Evidence from a Randomized Controlled Pilot Study». *Journal of Neuroengineering and Rehabilitation* 16, n.º 1 (15 de mayo de 2019): 56. https://doi.org/10.1186/s12984-019-0531-y.

[18] «Serveis a persones | adfo». Accedido 31 de mayo de 2019. https://www.adfo.cat/serveis/serveis-per-a-persones/.

[19] Shotton, Jamie, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, y Andrew Blake. «Real-Time Human Pose Recognition in Parts from Single Depth Images», s. f., 8.

[20] Steuer, Jonathan. «Defining Virtual Reality: Dimensions Determining Telepresence». *Journal of Communication* 42, n.º 4 (diciembre de 1992): 73-93. https://doi.org/10.1111/j.1460-2466.1992.tb00812.x.

[21] «TIPOS DE REALIDAD VIRTUAL». *REALIDAD VIRTUAL GBI* (blog), 6 de octubre de 2012. https://realidadvirtualgbi.wordpress.com/2012/10/06/sistemas-inmers/.

[22] «TIPOS DE REHABILITACIÓN – Clinica Recovery». Accedido 3 de junio de 2019. https://clinicarecovery.com/tipos-de-rehabilitacion/.

[23] Viñas-Diz, S., y M. Sobrido-Prieto. «[Virtual reality for therapeutic purposes in stroke: A systematic review]». *Neurologia (Barcelona, Spain)* 31, n.º 4 (mayo de 2016): 255-77. https://doi.org/10.1016/j.nrl.2015.06.012.

[24] «VIVE™ | Discover Virtual Reality Beyond Imagination». Accedido 24 de mayo de 2019. https://www.vive.com/eu/.

ANEXO A. Instrucciones Manejo de aplicación

Aquí explicamos brevemente los pasos a seguir para el/la usuario/a:

1. Montaje de la Kinect. Para empezar, debemos de conectar la Kinect con nuestro ordenador. En la figura 31 podemos observar cómo va conectada con el adaptador. Luego solo tendremos que conectar el puerto USB en nuestro ordenador y el enchufe en la fuente de alimentación AC. Coloque el sensor a una distancia entre 0,6m y 1,8m del suelo y en el borde de una superficie plana. Evitar colocarla en un sitio en el que la luz le dé directamente.

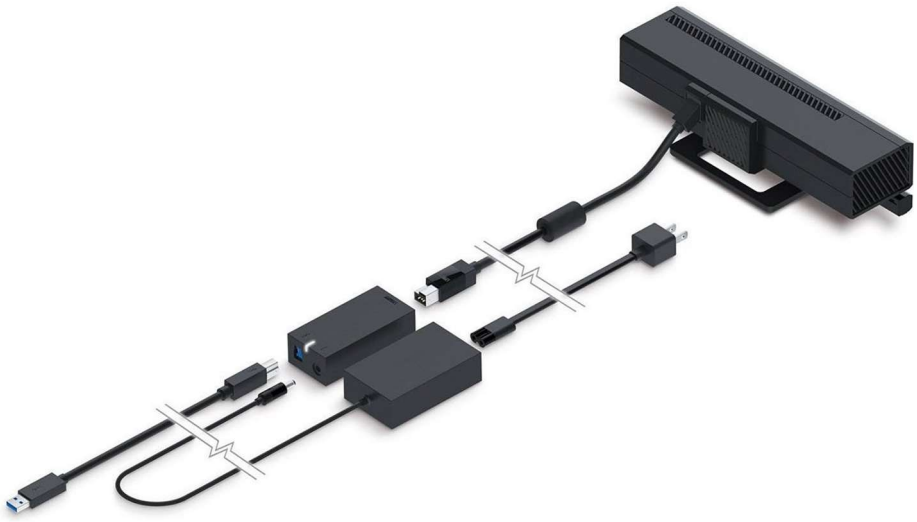


Figura 36. Kinect y sus componentes.

2. Enchufamos el USB del Arduino en el puerto del ordenador.

3. Ejecutaremos la aplicación arduino.exe, abriremos el código “Galgas.ino”, y pulsaremos el botón subir (ver figura 32.)

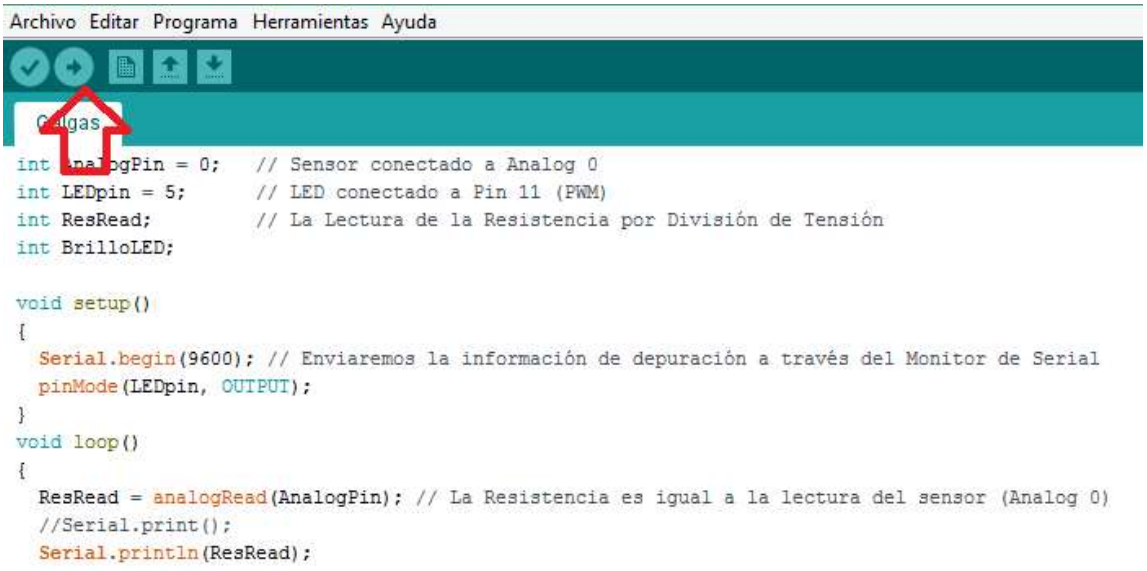


Figura 37. Código Arduino Galgas.

4. A continuación, tenemos que abrir nuestro proyecto con Unity. Pulsaremos el botón Play (ver figura 33).

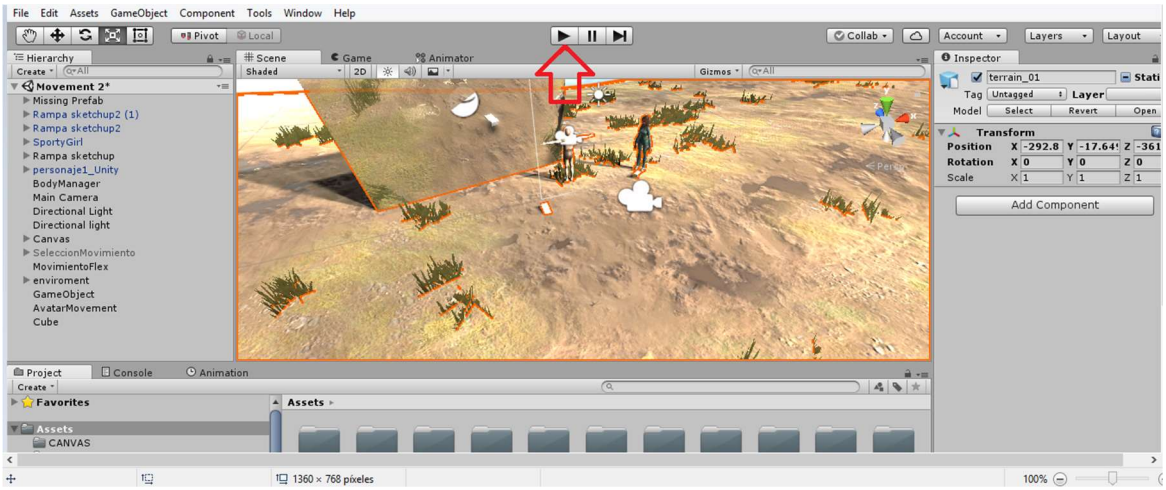


Figura 38. Pantalla Unity.

5. Seleccionar el ejercicio.

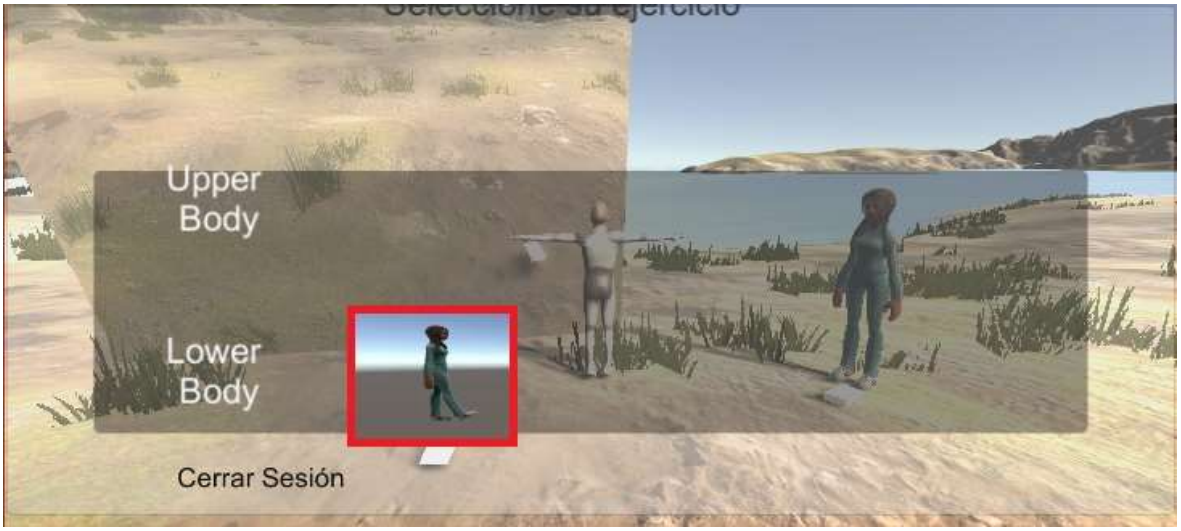


Figura 39. Selección del ejercicio.

6. Cuando el Semáforo esté verde, puede empezar a realizar el ejercicio. En las pestañas del recuadro superior a la derecha, puede seleccionar si quiere visualizar el valor de los ángulos actuales o máximos.

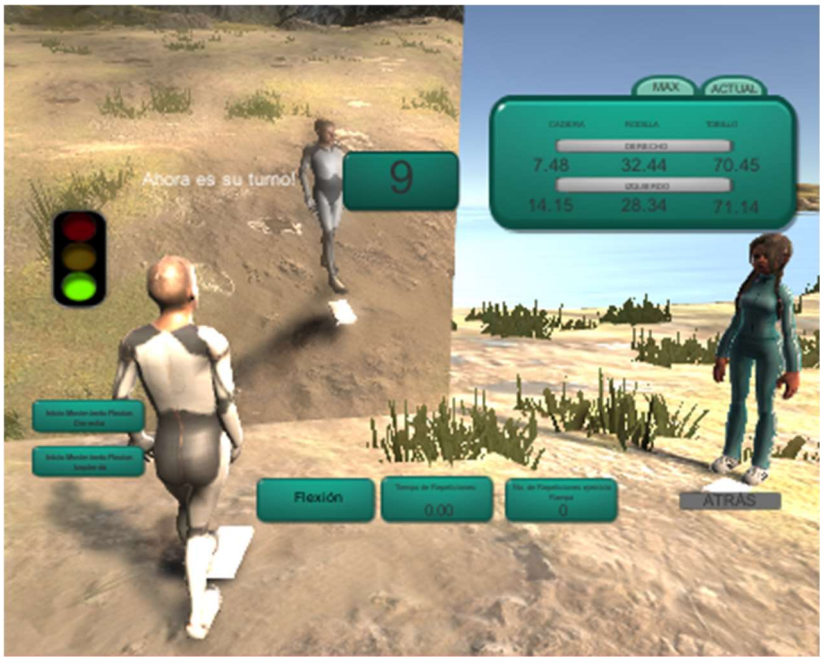


Figura 40. Realización del ejercicio.

ANEXO B. Programación del proyecto

B.1 BodySourceManager

```
using UnityEngine;
using System.Collections;
using Windows.Kinect;

public class BodySourceManager : MonoBehaviour
{
    private KinectSensor _Sensor;
    private BodyFrameReader _Reader;
    private Body[] _Data = null;


    public Body[] GetData()
    {
        return _Data;
    }

    public Windows.Kinect.Vector4 FloorPlanes
    {
        get;
        private set;
    }

    void Start ()
    {
        _Sensor = KinectSensor.GetDefault();

        if (_Sensor != null)
        {
            _Reader = _Sensor.BodyFrameSource.OpenReader();

            if (!_Sensor.IsOpen)
            {
                _Sensor.Open();
            }
        }
    }

    void Update ()
    {
        if (_Reader != null)
        {
            var frame = _Reader.AcquireLatestFrame();
            if (frame != null)
            {
                if (_Data == null)
                {
                    _Data = new Body[_Sensor.BodyFrameSource.BodyCount];
                }

                frame.GetAndRefreshBodyData(_Data);
```

```
                frame.Dispose();
                frame = null;
            }
        }
    }

    void OnApplicationQuit()
    {
        if (_Reader != null)
        {
            _Reader.Dispose();
            _Reader = null;
        }

        if (_Sensor != null)
        {
            if (_Sensor.IsOpen)
            {
                _Sensor.Close();
            }

            _Sensor = null;
        }
    }
}
```

B.2 BodySourceView

```
using UnityEngine;
using System.Collections;
using Windows.Data;
using System.Collections.Generic;
using Kinect = Windows.Kinect;
using Windows.Kinect;
using System.Linq;
using System.IO;
using System;
using UnityEngine.UI;


public class BodySourceView : MonoBehaviour
{
   public Text elbowPrint;
   public Text tiempoDeRepeticiones;
   public Resis_Sensor resis_Sensor;
   public Material BoneMaterial;
   public GameObject BodySourceManager;
   public GameObject _personaje1_Unity;
   public GameObject Master;
   public GameObject Reference;
   public GameObject Hips;
   public GameObject LeftUpLeg;
   public GameObject LeftLeg;
   public GameObject RightUpLeg;
   public GameObject RightLeg;
   public GameObject Spine;
   public GameObject Spine1;
   public GameObject Spine2;
   public GameObject LeftShoulder;
   public GameObject LeftArm;
   public GameObject LeftForeArm;
   public GameObject LeftHand;
   public GameObject Neck;
   public GameObject Head;
   public GameObject RightShoulder;
   public GameObject RightArm;
   public GameObject RightForeArm;
   public GameObject RightHand;
   public GameObject LeftFoot; //
   public GameObject RightFoot; //
   public GameObject RightAnkle; //
   public GameObject LeftAnkle; //
   public float F1 = 0;
   public float F2 = 0;
```

```
    public float F3 = 0;
    public float F4 = 0;
    public float F5 = 0;
    public float F6 = 0;
    public Text derecha = null; //Angulo Actual Derecho
    public Text izquierda = null; //Angulo Actual Izquierdo
    public Text derecha_1 = null;
    public Text izquierda_1 = null;
    public Text derecha_2 = null;
    public Text izquierda_2 = null;
    public float maxtot = 0;
    public int cont = -1;
    public bool rep = true;
    public Text angulo = null; //Angulo_Max_Derecho
    public Text angulo2 = null; //Angulo_Max_Izquierdo
    public Text angulo3 = null; //Angulo_Max_Derecho_Rodilla
    public Text angulo4 = null; //Angulo_Max_Izquierdo_Rodilla
    public Text angulo5 = null; //Angulo_Max_Derecho_Tobillo
    public Text angulo6 = null; //Angulo_Max_Izquierdo_Tobillo
    public bool angulonum = false;
    public GameObject semaforo;
    public GameObject semaforo2;
    public Text copiaej = null;
    public float valor;
    public float valor2;
    public float valor3;
    public float valor4;
    public float valor5;
    public float valor6;
    private Dictionary<ulong, GameObject> _Bodies = new Dictionary<ulong, GameObject>();
    private BodySourceManager _BodyManager;
    private string logFileName =
@"C:\Users\VISYON\Desktop\CO\A1\Movimientos_Miembros_Inferiores.txt";
    private string logFileHeader =
"ActualCadera,ActualRodilla,ActualTobillo,MaximoCadera,MaximoRodilla,MaximoTobillo";
    private float sampleRate = 1;
    private float logTimer;

// Canvax Maximos y Minimos value
/// </summary>

    public bool flex = false;
    public void ButtonClick()
    {
        Debug.Log("FLEX");
        flex = true;
        StartCoroutine(Seemaforo());
```

```
// Mapeo "puntos" del Cuerpo Humano

private Dictionary<Kinect.JointType, Kinect.JointType> _BoneMap = new
Dictionary<Kinect.JointType, Kinect.JointType>()
  {
    { Kinect.JointType.FootLeft, Kinect.JointType.AnkleLeft },
    { Kinect.JointType.AnkleLeft, Kinect.JointType.KneeLeft },
    { Kinect.JointType.KneeLeft, Kinect.JointType.HipLeft },
    { Kinect.JointType.HipLeft, Kinect.JointType.SpineBase },

    { Kinect.JointType.FootRight, Kinect.JointType.AnkleRight },
    { Kinect.JointType.AnkleRight, Kinect.JointType.KneeRight },
    { Kinect.JointType.KneeRight, Kinect.JointType.HipRight },
    { Kinect.JointType.HipRight, Kinect.JointType.SpineBase },

    { Kinect.JointType.HandTipLeft, Kinect.JointType.HandLeft },
    { Kinect.JointType.ThumbLeft, Kinect.JointType.HandLeft },
    { Kinect.JointType.HandLeft, Kinect.JointType.WristLeft },
    { Kinect.JointType.WristLeft, Kinect.JointType.ElbowLeft },
    { Kinect.JointType.ElbowLeft, Kinect.JointType.ShoulderLeft },
    { Kinect.JointType.ShoulderLeft, Kinect.JointType.SpineShoulder },

    { Kinect.JointType.HandTipRight, Kinect.JointType.HandRight },
    { Kinect.JointType.ThumbRight, Kinect.JointType.HandRight },
    { Kinect.JointType.HandRight, Kinect.JointType.WristRight },
    { Kinect.JointType.WristRight, Kinect.JointType.ElbowRight },
    { Kinect.JointType.ElbowRight, Kinect.JointType.ShoulderRight },
    { Kinect.JointType.ShoulderRight, Kinect.JointType.SpineShoulder },

    { Kinect.JointType.SpineBase, Kinect.JointType.SpineMid },
    { Kinect.JointType.SpineMid, Kinect.JointType.SpineShoulder },
    { Kinect.JointType.SpineShoulder, Kinect.JointType.Neck },
    { Kinect.JointType.Neck, Kinect.JointType.Head },
  };

  public void Start()
  {
    logTimer = sampleRate;
    File.AppendAllText(logFileName, logFileHeader + Environment.NewLine);

    _personaje1_Unity = GameObject.Find("personaje1_Unity");
    Master = GameObject.Find("master");
    Reference = GameObject.Find("Reference");
    Hips = GameObject.Find("Hips");
    LeftUpLeg = GameObject.Find("LeftUpLeg");
    LeftLeg = GameObject.Find("LeftLeg");
    RightUpLeg = GameObject.Find("RightUpLeg");
    RightLeg = GameObject.Find("RightLeg");
```

```
    Spine = GameObject.Find("Spine");
    Spine1 = GameObject.Find("Spine1");
    Spine2 = GameObject.Find("Spine2");
    LeftShoulder = GameObject.Find("LeftShoulder");
    LeftArm = GameObject.Find("LeftArm");
    LeftForeArm = GameObject.Find("LeftForeArm");
    LeftHand = GameObject.Find("LeftHand");
    Neck = GameObject.Find("Neck");
    Head = GameObject.Find("Head");
    RightShoulder = GameObject.Find("RightShoulder");
    RightArm = GameObject.Find("RightArm");
    RightForeArm = GameObject.Find("RightForeArm");
    RightHand = GameObject.Find("RightHand");
    RightAnkle = GameObject.Find("RightAnkle"); //
    LeftAnkle = GameObject.Find("LeftAnkle");   //
    RightFoot = GameObject.Find("RightFoot");  //
    LeftFoot = GameObject.Find("LeftFoot");    //

  }
  IEnumerator Seemaforo()
  {
    copiaej.text = "Realice el ejercicio propuesto";
    yield return new WaitForSeconds(7);
    Destroy(semaforo);
    copiaej.text = "Ahora es su turno!";
    semaforo2.SetActive(true);
  }

  IEnumerator Temporizador(float AngleFlex1co, float AngleFlexlcorrec, float AngleKneeR_Co,
float AngleKneeL_Co, float AngleAnkleR, float AngleAnkle)
  {
    flex = false;
    yield return new WaitForSeconds(0.5f);// Mira angulos FLex Cada segundo miembro inferior
    MAX(AngleFlex1co, AngleFlexlcorrec, AngleKneeR_Co, AngleKneeL_Co, AngleAnkleR,
AngleAnkle);
    flex = true;
  }

  public void MAX(float AngleFlex1co, float AngleFlexlcorrec, float AngleKneeR_Co, float
AngleKneeL_Co, float AngleAnkleR,float AngleAnklel)
  {
    if (AngleFlex1co > F1)
    {
      F1 = AngleFlex1co;
    }
    if (AngleFlexlcorrec > F2)
    {
      F2 = AngleFlexlcorrec;
```

```
    }
    if (AngleKneeR_Co > F3)
    {
      F3 = AngleKneeR_Co;
    }
    if (AngleKneeL_Co > F4)
    {
      F4 = AngleKneeR_Co;
    }
    if (AngleAnkleR > F5)
    {
      F5 = AngleAnkleR;
    }
    if (AngleAnklel > F6)
    {
      F6 = AngleAnklel;
    }
    valor = AngleFlex1co;
    valor2 = AngleFlexlcorrec;
    valor3 = AngleKneeR_Co;
    valor4 = AngleKneeL_Co;
    valor5 = AngleAnkleR;
    valor6 = AngleAnklel;
    derecha.text = (AngleFlex1co.ToString("0.00"));
    izquierda.text = (AngleFlexlcorrec.ToString("0.00"));
    derecha_1.text = (AngleKneeR_Co.ToString("0.00"));
    izquierda_1.text = (AngleKneeL_Co.ToString("0.00"));
    derecha_2.text = (AngleAnkleR.ToString("0.00"));
    izquierda_2.text = (AngleAnklel.ToString("0.00"));
    //Repeticiones(valor);
    angulo.text = (F1.ToString("0.00"));//
    angulo2.text = (F2.ToString("0.00"));
    angulo3.text = (F3.ToString("0.00"));
    angulo4.text = (F4.ToString("0.00"));
    angulo5.text = (F5.ToString("0.00"));
    angulo6.text = (F6.ToString("0.00"));
    putAnglesToFile();
  }

  void putAnglesToFile()
  {
    logTimer -= Time.deltaTime;
    if (logTimer < 0)
    {
      string ActualCadera = valor.ToString("0.00") + "," + valor2.ToString("0.00") + ",";
      string ActualRodilla = valor3.ToString("0.00") + "," + valor4.ToString("0.00") + ",";
      string ActualTobillo = valor5.ToString("0.00") + "," + valor6.ToString("0.00") + ",";
      string MaximoCadera = F1.ToString("0.00") + "," + F2.ToString("0.00") + ",";
```

```
        string MaximoRodilla = F3.ToString("0.00") + "," + F4.ToString("0.00") + ",";
        string MaximoTobillo = F5.ToString("0.00") + "," + F6.ToString("0.00") +
Environment.NewLine;
        string log = ActualCadera + ActualRodilla + ActualTobillo + MaximoCadera + MaximoCadera
+ MaximoRodilla + MaximoTobillo;
        File.AppendAllText(logFileName, log);
        logTimer = 1;
      }
    }

    void Update()
    {
      elbowPrint.text = resis_Sensor.getScore().ToString();
      tiempoDeRepeticiones.text = resis_Sensor.getRepetitionTime().ToString("0.00");

      if (BodySourceManager == null)
      {
        Debug.Log("BSC NULL");
        return;
      }

      _BodyManager = BodySourceManager.GetComponent<BodySourceManager>();
      if (_BodyManager == null)
      {
        Debug.Log("BSC NULL");
        return;
      }

      Kinect.Body[] data = _BodyManager.GetData();
      if (data == null)
      {
        return;
      }
      var floorPlane = _BodyManager.FloorPlanes;
      var newrotation = Quaternion.FromToRotation(new Vector3(floorPlane.X, floorPlane.Y,
floorPlane.Z), Vector3.up);

      List<ulong> trackedIds = new List<ulong>();
      foreach (var body in data)
      {
        if (body == null)
        {
          continue;
        }

        if (body.IsTracked)
        {
          trackedIds.Add(body.TrackingId);
```

```
        }
    }

    List<ulong> knownIds = new List<ulong>(_Bodies.Keys);

    // First delete untracked bodies
    foreach (ulong trackingId in knownIds)
    {
        if (!trackedIds.Contains(trackingId))
        {

            Destroy(_Bodies[trackingId]);
            _Bodies.Remove(trackingId);
        }
    }
    Debug.Log("body");
    foreach (var body in data)
    {

        if (body == null)
        {
            continue;
        }

        if (body.IsTracked)
        {
            if (!_Bodies.ContainsKey(body.TrackingId))
            {
                _Bodies[body.TrackingId] = CreateBodyObject(body.TrackingId);
            }

            RefreshBodyObject(body, _Bodies[body.TrackingId]);

            Quaternion SpineBase =
body.JointOrientations[JointType.SpineBase].Orientation.ChangeQuat(newrotation);
            Quaternion SpineMid =
body.JointOrientations[JointType.SpineMid].Orientation.ChangeQuat(newrotation);
            Quaternion SpineShoulder =
body.JointOrientations[JointType.SpineShoulder].Orientation.ChangeQuat(newrotation);
            Quaternion ShoulderLeft =
body.JointOrientations[JointType.ShoulderLeft].Orientation.ChangeQuat(newrotation);
            Quaternion ShoulderRight =
body.JointOrientations[JointType.ShoulderRight].Orientation.ChangeQuat(newrotation);
            Quaternion ElbowLeft =
body.JointOrientations[JointType.ElbowLeft].Orientation.ChangeQuat(newrotation);
            Quaternion WristLeft =
body.JointOrientations[JointType.WristLeft].Orientation.ChangeQuat(newrotation);
```

```
        Quaternion HandLeft = body.JointOrientations[JointType.HandLeft].Orientation.ChangeQuat(newrotation);
        Quaternion ElbowRight = body.JointOrientations[JointType.ElbowRight].Orientation.ChangeQuat(newrotation);
        Quaternion WristRight = body.JointOrientations[JointType.WristRight].Orientation.ChangeQuat(newrotation);
        Quaternion HandRight = body.JointOrientations[JointType.HandRight].Orientation.ChangeQuat(newrotation);
        Quaternion KneeLeft = body.JointOrientations[JointType.KneeLeft].Orientation.ChangeQuat(newrotation);
        Quaternion AnkleLeft = body.JointOrientations[JointType.AnkleLeft].Orientation.ChangeQuat(newrotation);
        Quaternion KneeRight = body.JointOrientations[JointType.KneeRight].Orientation.ChangeQuat(newrotation);
        Quaternion AnkleRight = body.JointOrientations[JointType.AnkleRight].Orientation.ChangeQuat(newrotation);
        Quaternion FootRight = body.JointOrientations[JointType.FootRight].Orientation.ChangeQuat(newrotation); //
        Quaternion FootLeft = body.JointOrientations[JointType.FootLeft].Orientation.ChangeQuat(newrotation); //
        FootRight.y = 1;
        FootRight.w = 0;
        FootLeft.y = 1;
        FootLeft.w = 0;
        Spine1.transform.rotation = SpineMid * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
        RightArm.transform.rotation = ElbowRight * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
        RightForeArm.transform.rotation = WristRight * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
        RightHand.transform.rotation = HandRight * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
        LeftArm.transform.rotation = ElbowLeft * Quaternion.AngleAxis(180, new Vector3(0, 1, 0)) *
          Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
        LeftForeArm.transform.rotation = WristLeft * Quaternion.AngleAxis(180, new Vector3(0, 1, 0)) *
          Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
        RightFoot.transform.rotation = FootRight * Quaternion.AngleAxis(0, new Vector3(0, 0, 1)) * Quaternion.AngleAxis(-90, new Vector3(0, 1, 0)); // ojo -90
        LeftFoot.transform.rotation = FootLeft * Quaternion.AngleAxis(180, new Vector3(1, 0, 0)) * Quaternion.AngleAxis(-90, new Vector3(0, 1, 0));//
        LeftHand.transform.rotation = HandLeft * Quaternion.AngleAxis(180, new Vector3(0, 1, 0)) * Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
        RightUpLeg.transform.rotation = KneeRight * Quaternion.AngleAxis(90, new Vector3(0, 1, 0)) * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
        RightLeg.transform.rotation = AnkleRight * Quaternion.AngleAxis(90, new Vector3(0, 1, 0)) * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
```

```
        LeftUpLeg.transform.rotation = KneeLeft * Quaternion.AngleAxis(90, new Vector3(0, 1, 0))
* Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
        LeftLeg.transform.rotation = AnkleLeft * Quaternion.AngleAxis(90, new Vector3(0, 1, 0)) *
Quaternion.AngleAxis(90, new Vector3(0, 0, 1));

        var moveposition = body.Joints[JointType.SpineMid].Position;
        Master.transform.position = new Vector3(moveposition.X, moveposition.Y, -
moveposition.Z);


                        //////////////MiembroInferior/////////////////

        // :::::::::::::::::::   FLEXION  Hip :::::::::::::::::::: //
        if (flex = true)
        {
          //Variables Hip
          float x11 = body.Joints[JointType.HipRight].Position.X;
          float y11 = body.Joints[JointType.HipRight].Position.Y;
          float z11 = body.Joints[JointType.HipRight].Position.Z;
          float x22 = body.Joints[JointType.HipLeft].Position.X;
          float y22 = body.Joints[JointType.HipLeft].Position.Y;
          float z22 = body.Joints[JointType.HipLeft].Position.Z;
          float x33 = body.Joints[JointType.SpineMid].Position.X;
          float y33 = body.Joints[JointType.SpineMid].Position.Y;
          float z33 = body.Joints[JointType.SpineMid].Position.Z;
          float k11 = body.Joints[JointType.KneeLeft].Position.X;
          float k22 = body.Joints[JointType.KneeLeft].Position.Y;
          float k33 = body.Joints[JointType.KneeLeft].Position.Z;
          float k1l = body.Joints[JointType.KneeRight].Position.X;
          float k2l = body.Joints[JointType.KneeRight].Position.Y;
          float k3l = body.Joints[JointType.KneeRight].Position.Z;
          float u11 = body.Joints[JointType.SpineBase].Position.X;
          float u22 = body.Joints[JointType.SpineBase].Position.Y;
          float u33 = body.Joints[JointType.SpineBase].Position.Z;


          //plane1  (SpineMid-HipRight)
          Vector3 vector1 = new Vector3(x22 - x11, y22 - y11, z22 - z11); // Vectorial Result
HipRight
          Vector3 vector2 = new Vector3(x33 - x11, y33 - y11, z33 - z11);
          Vector3 Cp1;
          Cp1 = Vector3.Cross(vector1, vector2); //Producto entre el vector 1 y 2
          Vector3 normalvect = new Vector3(Cp1.x, Cp1.y, Cp1.z); //Vector normal
          Vector3 Vec_Aux1 = new Vector3(x33 - u11, y33 - u22, z33 - u33);
          Vector3 newcp;
          newcp = Vector3.Cross(normalvect, Vec_Aux1);  //Vector resultante = vector normal,
vector entre los dos puntos
```

```
        float daux1 = (newcp.x * (-x33) + newcp.y * (-y33) + newcp.z * (-z33)); //Magnitud del vector resultantE

        //Right side projection and result Knee and Hip
        float lambda_2 = (((-newcp.x) * k1l + (-newcp.y) * k2l + (-newcp.z) * k3l) - daux1) / ((newcp.x) * (newcp.x) + (newcp.y) * (newcp.y) + (newcp.z) * (newcp.z));
        Vector3 KneeProject = new Vector3(k1l + (lambda_2 * newcp.x), k2l + (lambda_2 * newcp.y), k3l + (lambda_2 * newcp.z));
        float KneeXr = body.Joints[JointType.ShoulderRight].Position.X; //Impresión en los tres ejes (x,y,z) de la rodilla
        float KneeYr = body.Joints[JointType.ShoulderRight].Position.Y;
        float KneeZr = body.Joints[JointType.ShoulderRight].Position.Z;
        float lambda_aux1 = (((-newcp.x) * KneeXr + (-newcp.y) * KneeYr + (-newcp.z) * KneeZr) - daux1) / ((newcp.x) * (newcp.x) + (newcp.y) * (newcp.y) + (newcp.z) * (newcp.z));
        Vector3 Hipproject = new Vector3(KneeXr + (lambda_aux1 * newcp.x), KneeYr + (lambda_aux1 * newcp.y), KneeZr + (lambda_aux1 * newcp.z));
        Vector3 VectorKnee1 = new Vector3(Hipproject.x - KneeProject.x, Hipproject.y - KneeProject.y, Hipproject.z - KneeProject.z);
        float AngleFlex1 = Vector3.Angle(Vec_Aux1, VectorKnee1);
        float AngleFlex1co = AngleFlex1; //Angulo de flexión

        //left side Knee and Hip
        float lambda_3 = (((-newcp.x) * k11 + (-newcp.y) * k22 + (-newcp.z) * k33) - daux1) / ((newcp.x) * (newcp.x) + (newcp.y) * (newcp.y) + (newcp.z) * (newcp.z));
        Vector3 Kneeproject = new Vector3(k11 + (lambda_3 * newcp.x), k22 + (lambda_3 * newcp.y), k33 + (lambda_3 * newcp.z));
        float KneeXl = body.Joints[JointType.ShoulderRight].Position.X; //Impresión en los tres ejes (x,y,z) de la rodilla
        float KneeYl = body.Joints[JointType.ShoulderRight].Position.Y;
        float KneeZl = body.Joints[JointType.ShoulderRight].Position.Z;
        float lambda_4 = (((-newcp.x) * KneeXl + (-newcp.y) * KneeYl + (-newcp.z) * KneeZl) - daux1) / ((newcp.x) * (newcp.x) + (newcp.y) * (newcp.y) + (newcp.z) * (newcp.z));
        Vector3 Hipproj = new Vector3(KneeXl + (lambda_4 * newcp.x), KneeYl + (lambda_4 * newcp.y), KneeZl + (lambda_4 * newcp.z));
        Vector3 Hip_A = new Vector3(Kneeproject.x - Hipproj.x, Kneeproject.y - Hipproj.y, Kneeproject.z - Hipproj.z);
        float AngleFlexl = Vector3.Angle(Vec_Aux1, Hip_A);
        float AngleFlexlcorrec = 180 - AngleFlexl; // Se le resta a 180 los grados que indique la operación
        //StartCoroutine(Temporizador(AngleFlex1co, AngleFlexlcorrec));


        ////////////////////////// Knee ///////////////////////////////////

        float HipXr = body.Joints[JointType.HipRight].Position.X;
        float HipYr = body.Joints[JointType.HipRight].Position.Y;
        float HipZr = body.Joints[JointType.HipRight].Position.Z;
        float KneeeX = body.Joints[JointType.KneeRight].Position.X;
```

```
            float KneeeY = body.Joints[JointType.KneeRight].Position.Y;
            float KneeeZ = body.Joints[JointType.KneeRight].Position.Z;
            float AnkleXr = body.Joints[JointType.AnkleRight].Position.X;
            float AnkleYr = body.Joints[JointType.AnkleRight].Position.Y;
            float AnkleZr = body.Joints[JointType.AnkleRight].Position.Z;
            float HipXl = body.Joints[JointType.HipLeft].Position.X;
            float HipYl = body.Joints[JointType.HipLeft].Position.Y;
            float HipZl = body.Joints[JointType.HipLeft].Position.Z;
            float lelKneeX = body.Joints[JointType.KneeLeft].Position.X;
            float lelKneeY = body.Joints[JointType.KneeLeft].Position.Y;
            float lelKneeZ = body.Joints[JointType.KneeLeft].Position.Z;
            float AnkleXl = body.Joints[JointType.AnkleLeft].Position.X;
            float AnkleYl = body.Joints[JointType.AnkleLeft].Position.Y;
            float AnkleZl = body.Joints[JointType.AnkleLeft].Position.Z;

            //right side results Knee
            Vector3 VecHKR = new Vector3(HipXr - KneeeX, HipYr - KneeeY, HipZr - KneeeZ); // Vector cadera- rodilla
            Vector3 VecKWR = new Vector3(AnkleXr - KneeeX, AnkleYr - KneeeY, AnkleZr - KneeeZ); // Vector rodilla- tobillo
            float AngleKneeR = Vector3.Angle(VecHKR, VecKWR); //Vector resultante
            float AngleKneeR_Co = 180 - AngleKneeR;
            //Debug.Log(" Angle right Knee: " + AngleKneeR_Co); // Impresión del vector-- lo muestra angulo flexión de la rodilla

            //left side results Knee
            Vector3 VecHKL = new Vector3(HipXl - lelKneeX, HipYl - lelKneeY, HipZl - lelKneeZ);
            Vector3 VecHWL = new Vector3(AnkleXl - lelKneeX, AnkleYl - lelKneeY, AnkleZl - lelKneeZ);
            float AngleKneeL = Vector3.Angle(VecHKL, VecHWL);
            float AngleKneeL_Co = 180 - AngleKneeL;


            // ::::::::::::::::::::   ANKLE EXTENSION   :::::::::::::::::::: //

            float FootXr = body.Joints[JointType.FootRight].Position.X;
            float FootYr = body.Joints[JointType.FootRight].Position.Y;
            float FootZr = body.Joints[JointType.FootRight].Position.Z;
            float FootXl = body.Joints[JointType.FootLeft].Position.X;
            float FootYl = body.Joints[JointType.FootLeft].Position.Y;
            float FootZl = body.Joints[JointType.FootLeft].Position.Z;


            //Right side results
            Vector3 VecAnkleFootr = new Vector3(FootXr - AnkleXr, FootYr - AnkleYr, FootZr - AnkleZr); //Vector tobillo- pie
            Vector3 VecKNFR = new Vector3(KneeXr - AnkleXr, KneeYr - AnkleYr, KneeZr - AnkleZr); //Vector de rodilla-Pie
```

```
            float AngleAnkleR = 180 - Vector3.Angle(VecAnkleFootr, VecKNFR); //Angulo del vector
del tobillo right

            //eft side results
            Vector3 VecAnkleFootl = new Vector3(FootXl - AnkleXl, FootYl - AnkleYl, FootZl -
AnkleZl);
            Vector3 VecKNFl = new Vector3(KneeXl - AnkleXl, KneeYl - AnkleYl, KneeZl - AnkleZl);
            float AngleAnklel = 180- Vector3.Angle(VecAnkleFootl, VecKNFl);

            Debug.Log("Temporizador");
            StartCoroutine(Temporizador(AngleFlex1co, AngleFlexlcorrec, AngleKneeR_Co,
AngleKneeL_Co, AngleAnkleR, AngleAnklel));


          }

        }
    }

  }


  private GameObject CreateBodyObject(ulong id)
  {
    GameObject body = new GameObject("Body:" + id);

    for (Kinect.JointType jt = Kinect.JointType.SpineBase; jt <= Kinect.JointType.ThumbRight; jt++)
    {
      GameObject jointObj = GameObject.CreatePrimitive(PrimitiveType.Cube);

      LineRenderer lr = jointObj.AddComponent<LineRenderer>();
      lr.SetVertexCount(2);
      lr.material = BoneMaterial;
      lr.SetWidth(0.0f, 0.0f);

      jointObj.transform.localScale = new Vector3(0.0f, 0.0f, 0.0f);
      jointObj.name = jt.ToString();
      jointObj.transform.parent = body.transform;
    }

    return body;

  }

  private void RefreshBodyObject(Kinect.Body body, GameObject bodyObject)
  {
    for (Kinect.JointType jt = Kinect.JointType.SpineBase; jt <= Kinect.JointType.ThumbRight; jt++)
    {
```

```
            Kinect.Joint sourceJoint = body.Joints[jt];
            Kinect.Joint? targetJoint = null;

            if (_BoneMap.ContainsKey(jt))
            {
                targetJoint = body.Joints[_BoneMap[jt]];
            }

            Transform jointObj = bodyObject.transform.Find(jt.ToString());
            jointObj.localPosition = GetVector3FromJoint(sourceJoint);

            LineRenderer lr = jointObj.GetComponent<LineRenderer>();
            if (targetJoint.HasValue)
            {
                lr.SetPosition(0, jointObj.localPosition);
                lr.SetPosition(1, GetVector3FromJoint(targetJoint.Value));
                lr.SetColors(GetColorForState(sourceJoint.TrackingState),
GetColorForState(targetJoint.Value.TrackingState));
            }
            else
            {
                lr.enabled = false;
            }
        }
    }

    private static Color GetColorForState(Kinect.TrackingState state)
    {
        switch (state)
        {
            case Kinect.TrackingState.Tracked:
                return Color.green;

            case Kinect.TrackingState.Inferred:
                return Color.red;

            default:
                return Color.black;
        }
    }

    private static Vector3 GetVector3FromJoint(Kinect.Joint joint)
    {
        return new Vector3(joint.Position.X * 10, joint.Position.Y * 10, joint.Position.Z * 10);
    }
}
```

B.3 Resis_Sensor

```
using UnityEngine;
using System;
using System.IO.Ports; //incluimos el namespace Sustem.IO.Ports
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using System.IO;

public class Resis_Sensor : MonoBehaviour
{

    SerialPort serialPort = new SerialPort("COM9", 9600); //Inicializamos el puerto
serie
    //SerialPort serialPort = new SerialPort();
    private int score = 0;
    private int cont = 0;
    public float TiempoRepeticiones = 0;
    private bool startCounting = false;
    private string logFileName =
@"C:\Users\VISYON\Desktop\CO\A1\Tiempo_Repeticiones.txt";
    private string logFileHeader = "TiempoDeRepeticiones";
    private string externalAngleLog =
@"C:\Users\VISYON\Desktop\CO\A1\Movimientos_Miembros_Inferiores.txt";
    private string repetitionSeparator = "361.00,0.00,0.00,0.00,0.00,0.00";

    void resetAndStartCountingTime()
    {
        putDataInFile();
        TiempoRepeticiones = 0;
        startCounting = true;
    }

    void putDataInFile()
    {
        File.AppendAllText(logFileName, TiempoRepeticiones.ToString("0.00") +
Environment.NewLine);
        File.AppendAllText(externalAngleLog, repetitionSeparator +
Environment.NewLine);

    }
    void Start()
    {
        Debug.Log("Counter starts");
        Debug.Log(this);
        File.AppendAllText(logFileName, logFileHeader + Environment.NewLine);
        serialPort.Open(); //Abrimos una nueva conexión de puerto serie
        serialPort.ReadTimeout = 1; //Establecemos el tiempo de espera cuando una
operación de lectura no finaliza
    }

    public int getScore()
    {
        return score;
    }
```

```
    public float getRepetitionTime()
    {
        return TiempoRepeticiones;
    }
    void Update()
    {
        if (serialPort.IsOpen) //comprobamos que el puerto esta abierto
        {
            if (startCounting)
            {
                TiempoRepeticiones += Time.deltaTime;
            }
            try //utilizamos el bloque try/catch para detectar una posible
excepción.
            {

                string value = serialPort.ReadLine(); //leemos una linea del puerto
serie y la almacenamos en un string
                print(value); //impresion de la linea leida para verificar que
leemos el dato que manda nuestro Arduino
                              //string[] vec6 = value.Split(','); //Separamos el
String leido valiendonos
                              //                                  //de las comas y
almacenamos los valores en un array.
                int val = int.Parse(value);

                if (val > 0)
                {

                    if (cont == 0)
                    {
                        Debug.Log("cont == 0");
                        resetAndStartCountingTime();
                        score = score + 1;
                    };
                    cont = 1;
                }
                else
                {
                    Debug.Log("cont = 0 assignement");
                    cont = 0;
                };
                print("score=" + score);
                //Debug.Log("Tiempo de la repeticion =" + TiempoRepeticiones);
            }

            catch (System.Exception ex)
            {
                ex = new System.Exception();

            }
        }

    }


}
```

B.4 Timer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Timer : MonoBehaviour {
    public Text Contador_1;
    public float Tiempo = 10f;
    private bool started = false;
       // Use this for initialization
       void Start () {
        Contador_1.text = "" + Tiempo;
       }

       // Update is called once per frame
       void Update () {
        if(started && Tiempo > 0) {
            Tiempo -= Time.deltaTime;
            Contador_1.text = "" + Tiempo.ToString("f0");
        }

        //Debug.Log("Timer: update");
       }

    public void ButtonClick()
    {
        StartCoroutine(waitAndCount());

    }

    IEnumerator waitAndCount()
    {
        yield return new WaitForSeconds(7);
        started = true;

    }
}
```

B.5 Vector4

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public static class Vector4
{
    public static Quaternion ChangeQuat(this Windows.Kinect.Vector4 newvector,
                                                                    Quaternion
newrotation)
    {
        return Quaternion.Inverse(newrotation) *
                    new Quaternion(-newvector.X, -newvector.Y, newvector.Z,
newvector.W);
    }
```

B.6 LevantarPierna

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(Animator))]
public class LevantarPierna : MonoBehaviour
{

    private Animator animator;

    const int countOfDamageAnimations = 3;
    int lastDamageAnimation = -1;

    public void ButtonClick()
    {

        animator.SetBool("Squat", false);
        animator.SetFloat("Speed", 0f);
        animator.SetBool("Aiming", false);
        animator.SetTrigger("Izquierda_Pierna");
    }
    void Awake()
    {
        animator = GetComponent<Animator>();
    }

}
```

B.7 LevantarPierna_Iz

```
using UnityEngine;
using System.Collections;

[RequireComponent(typeof(Animator))]
public class LevantarPierna_Iz : MonoBehaviour
{

    private Animator animator;

    const int countOfDamageAnimations = 3;
    int lastDamageAnimation = -1;

    public void ButtonClick()
    {

        animator.SetBool("Squat", false);
        animator.SetFloat("Speed", 0f);
        animator.SetBool("Aiming", false);
        animator.SetTrigger("Derecha_Pierna");
    }
    void Awake()
    {
        animator = GetComponent<Animator>();
    }

}
```

B.8 Galgas

```
int AnalogPin = 0;   // Sensor conectado a Analog 0
int LEDpin = 6;      // LED conectado a Pin 11 (PWM)
int ResRead;         // La Lectura de la Resistencia por División de Tensión
int BrilloLED;

void setup()
{
 Serial.begin(9600); // Enviaremos la información de depuración a través del Monitor de Serial
 pinMode(LEDpin, OUTPUT);
}
void loop()
{
 ResRead = analogRead(AnalogPin); // La Resistencia es igual a la lectura del sensor (Analog 0)
 Serial.print("Lectura Analogica = ");
 Serial.println(ResRead);


 BrilloLED = map(ResRead, 0, 1023, 0, 255);
// Cambiar el rango de la lectura analógica (0-1023)
// Utilizamos en analogWrite 8 bits (0-255) configurados en el map
if (BrilloLED>2){
 Serial.println("High");
 digitalWrite(LEDpin, HIGH);}
else {
 digitalWrite(LEDpin, LOW);}

 delay(100); //Cien “ms” de espera en cada lectura
}
```

ANEXO C. Planificación del proyecto – Diagrama de Gantt

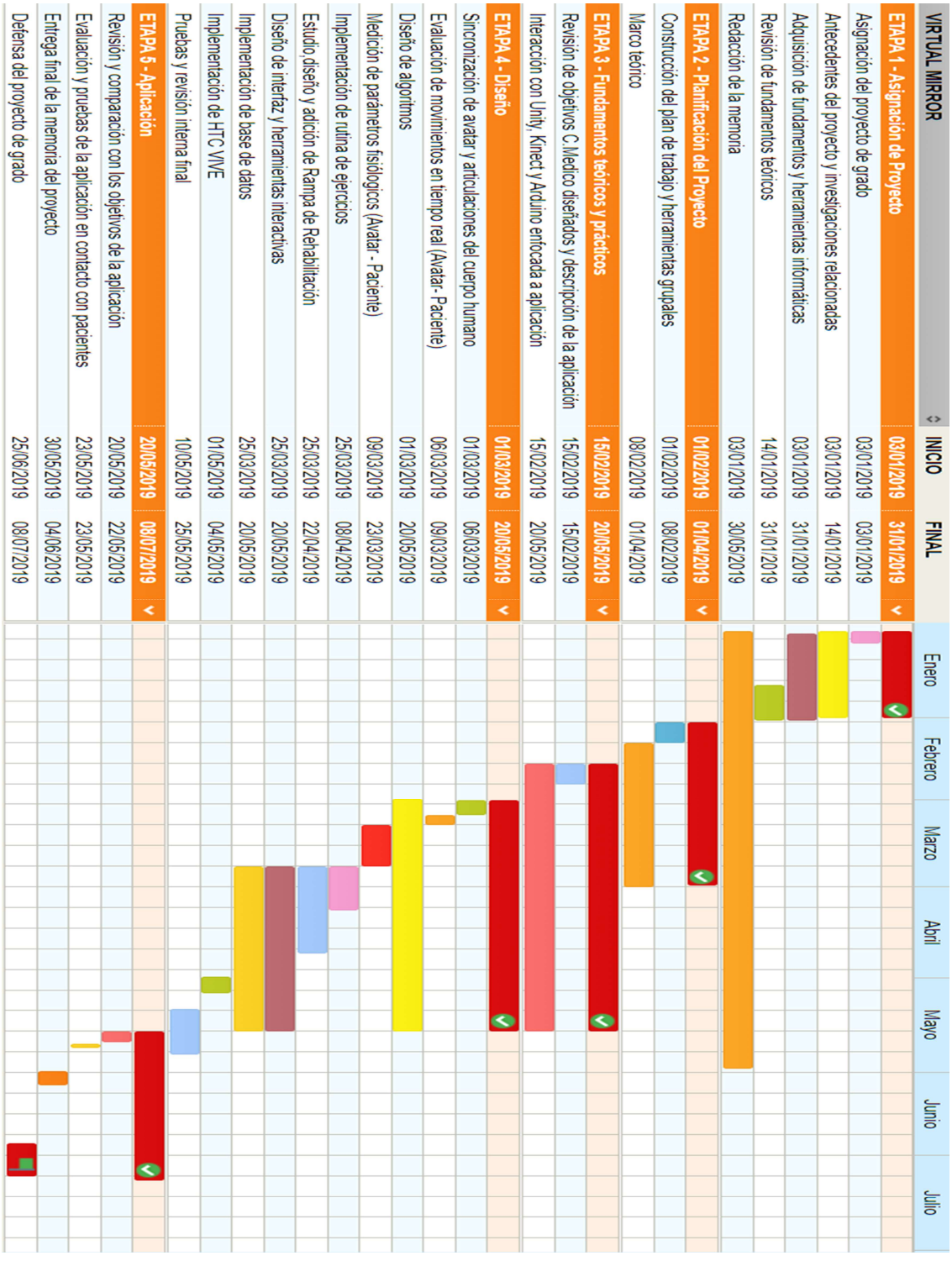


Figura 36. Diagrama de Gantt del proyecto (Extendido).